Application Kit Framework Reference

2009-08-28

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Contents

Figures, Tables, and Listings

The Application Kit

Framework	/System/Library/Frameworks/AppKit.framework
Header file directories	/System/Library/Frameworks/AppKit.framework/Headers
Declared in	AppKitErrors.h NSATSTypesetter.h NSAccessibility.h NSActionCell.h NSAffineTransform.h NSAlert.h NSAnimation.h NSAnimationContext.h NSAppleScriptExtensions.h NSApplication.h NSApplicationScripting.h NSArrayController.h NSAttributedString.h NSBezierPath.h NSBitmapImageRep.h NSBox.h NSBrowser.h NSBrowserCell.h NSButton.h NSButtonCell.h NSCIImageRep.h NSCachedImageRep.h NSCell.h NSClipView.h NSCollectionView.h NSColor.h NSColorList.h NSColorPanel.h NSColorPicker.h NSColorPicking.h NSColorSpace.h NSColorWell.h NSComboBox.h NSComboBoxCell.h NSControl.h NSController.h NSCursor.h NSCustomImageRep.h NSDatePicker.h NSDatePickerCell.h NSDictionaryController.h NSDockTile.h NSDocument.h

NSDocumentController.h
NSDocumentScripting.h
NSDragging.h
NSDrawer.h
NSEPSImageRep.h
NSErrors.h
NSEvent.h
NSFileWrapper.h
NSFont.h
NSFontDescriptor.h
NSFontManager.h
NSFontPanel.h
NSForm.h
NSFormCell.h
NSGlyphGenerator.h
NSGlyphInfo.h
NSGradient.h
NSGraphics.h
NSGraphicsContext.h
NSHelpManager.h
NSImage.h
NSImageCell.h
NSImageRep.h
NSImageView.h
NSInputManager.h
NSInterfaceStyle.h
NSKeyValueBinding.h
NSLayoutManager.h
NSLevelIndicator.h
NSLevelIndicatorCell.h
NSMatrix.h
NSMenu.h
NSMenuItem.h
NSMenuItemCell.h
NSMenuView.h
NSNib.h
NSNibConnector.h
NSNibControlConnector.h
NSNibDeclarations.h
NSNibLoading.h
NSNibOutletConnector.h
NSObjectController.h
NSOpenGL.h
NSOpenGLLayer.h
NSOpenGLView.h
NSOpenPanel.h
NSOutlineView.h
NSPDFImageRep.h
NSPICTImageRep.h
NSPageLayout.h
NSPanel.h
NSParagraphStyle.h
NSPasteboard.h

NSPasteboardItem.h
NSPathCell.h
NSPathComponentCell.h
NSPathControl.h
NSPersistentDocument.h
NSPopUpButton.h
NSPopUpButtonCell.h
NSPredicateEditor.h
NSPredicateEditorRowTemplate.h
NSPrintInfo.h
NSPrintOperation.h
NSPrintPanel.h
NSPrinter.h
NSProgressIndicator.h
NSResponder.h
NSRuleEditor.h
NSRulerMarker.h
NSRulerView.h
NSRunningApplication.h
NSSavePanel.h
NSScreen.h
NSScrollView.h
NSScroller.h
NSSearchField.h
NSSearchFieldCell.h
NSSecureTextField.h
NSSegmentedCell.h
NSSegmentedControl.h
NSShadow.h
NSSimpleHorizontalTypesetter.h
NSSlider.h
NSSliderCell.h
NSSound.h
NSSpeechRecognizer.h
NSSpeechSynthesizer.h
NSSpellChecker.h
NSSpellProtocol.h
NSSplitView.h
NSStatusBar.h
NSStatusItem.h
NSStepper.h
NSStepperCell.h
NSStringDrawing.h
NSTabView.h
NSTabViewItem.h
NSTableColumn.h
NSTableHeaderCell.h
NSTableHeaderView.h
NSTableView.h
NSText.h
NSTextAttachment.h
NSTextContainer.h
NSTextField.h

NSTextFieldCell.h
NSTextInputClient.h
NSTextInputContext.h
NSTextList.h
NSTextStorage.h
NSTextStorageScripting.h
NSTextTable.h
NSTextView.h
NSTokenField.h
NSTokenFieldCell.h
NSToolbar.h
NSToolbarItem.h
NSToolbarItemGroup.h
NSTouch.h
NSTrackingArea.h
NSTreeController.h
NSTreeNode.h
NSTypesetter.h
NSUserDefaultsController.h
NSUserInterfaceItemSearching.h
NSUserInterfaceValidation.h
NSView.h
NSViewController.h
NSWindow.h
NSWindowController.h
NSWindowScripting.h
NSWorkspace.h

Introduction

The Application Kit is a framework containing all the objects you need to implement your graphical, event-driven user interface: windows, panels, buttons, menus, scrollers, and text fields. The Application Kit handles all the details for you as it efficiently draws on the screen, communicates with hardware devices and screen buffers, clears areas of the screen before drawing, and clips views. The number of classes in the Application Kit may seem daunting at first. However, most Application Kit classes are support classes that you use indirectly. You also have the choice at which level you use the Application Kit:

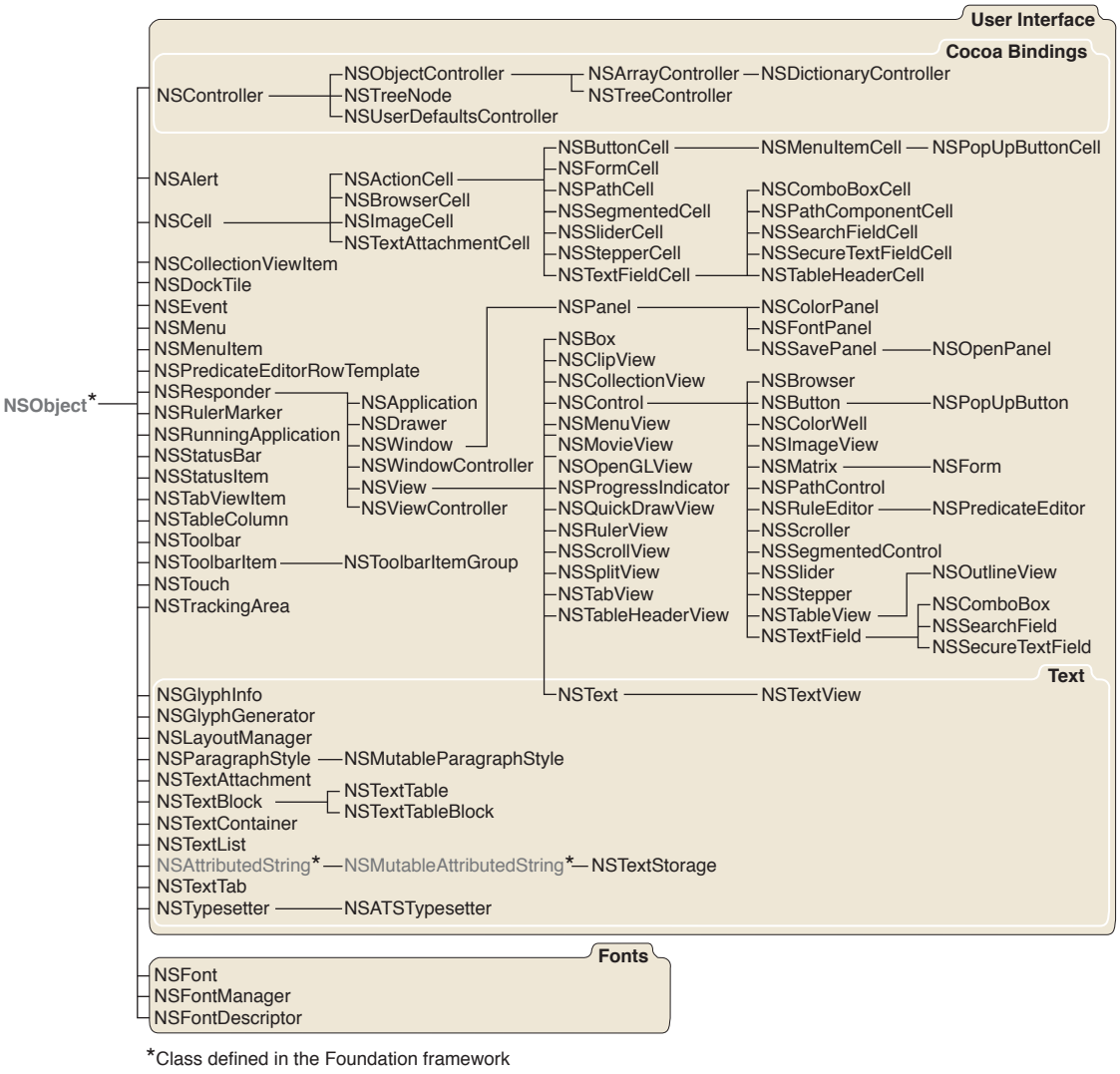
- Use Interface Builder to create connections from user interface objects to your application objects. In this case, all you need to do is implement your application classes—implement those action and delegate methods. For example, implement the method that is invoked when the user selects a menu item.
- Control the user interface programmatically, which requires more familiarity with Application Kit classes and protocols. For example, allowing the user to drag an icon from one window to another requires some programming and familiarity with the `NSDragging...` protocols.
- Implement your own objects by subclassing NSView or other classes. When subclassing NSView you write your own drawing methods using graphics functions. Subclassing requires a deeper understanding of how the Application Kit works.

To learn more about the Application Kit, review the `NSApplication`, `NSWindow`, and `NSView` class specifications, paying close attention to delegate methods. For a deeper understanding of how the Application Kit works, see the specifications for `NSResponder` and `NSRunLoop` (`NSRunLoop` is in the Foundation framework).

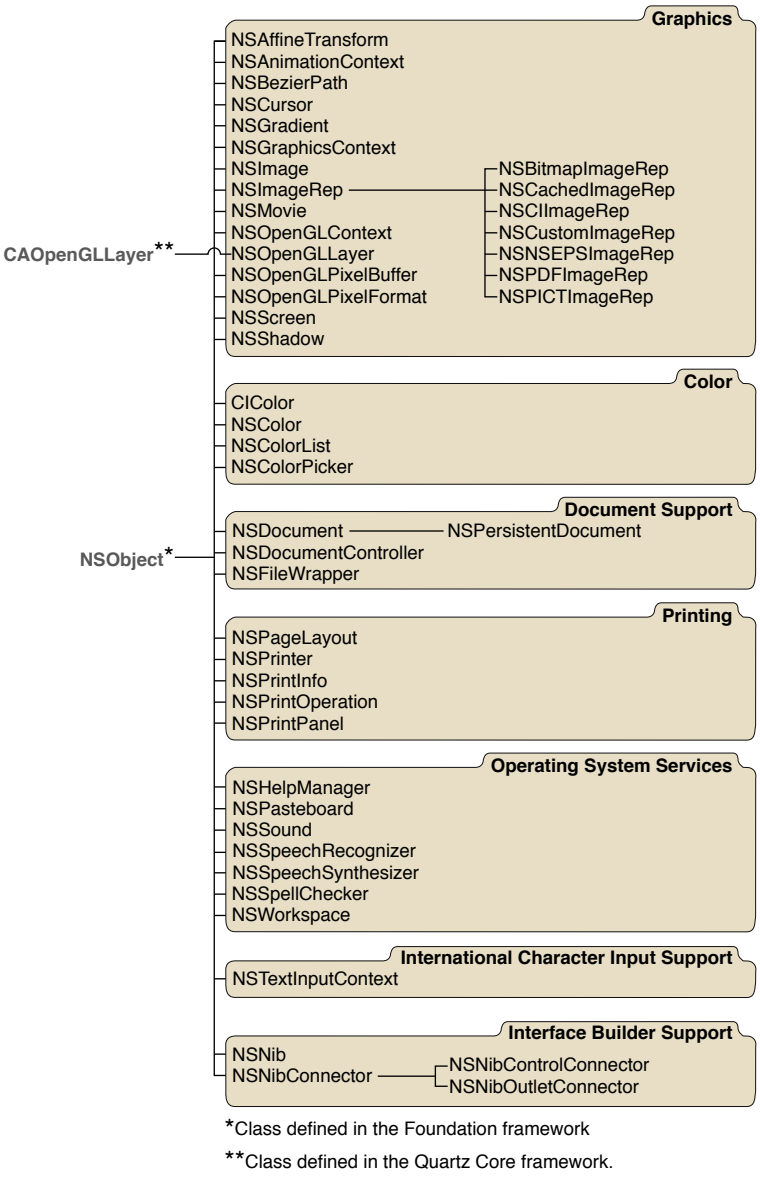
Application Kit Classes and Protocols

The Application Kit is large; it comprises more than 125 classes and protocols. The classes all descend from the Foundation framework's NSObject class (see Figure I-1 (page 47)). The following sections briefly describe some of the topics that the Application Kit addresses through its classes and protocols.

Figure I-1 Cocoa Objective-C Class Hierarchy for Application Kit



Objective-C Application Kit Continued



*Class defined in the Foundation framework

**Class defined in the Quartz Core framework.

Encapsulating an Application

Every application uses a single instance of `NSApplication` to control the main event loop, keep track of the application's windows and menus, distribute events to the appropriate objects (that is, itself or one of its windows), set up autorelease pools, and receive notification of application-level events. An `NSApplication` object has a delegate (an object that you assign) that is notified when the application starts or terminates, is hidden or activated, should open a file selected by the user, and so forth. By setting the `NSApplication` object's delegate and implementing the delegate methods, you customize the behavior of your application without having to subclass `NSApplication`.

General Event Handling and Drawing

The `NSResponder` class defines the responder chain, an ordered list of objects that respond to user events. When the user clicks the mouse button or presses a key, an event is generated and passed up the responder chain in search of an object that can "respond" to it. Any object that handles events must inherit from the `NSResponder` class. The core Application Kit classes, `NSApplication`, `NSWindow`, and NSView, inherit from `NSResponder`.

An `NSApplication` object maintains a list of `NSWindow` objects—one for each window belonging to the application—and each `NSWindow` object maintains a hierarchy of NSView objects. The view hierarchy is used for drawing and handling events within a window. An `NSWindow` object handles window-level events, distributes other events to its views, and provides a drawing area for its views. An `NSWindow` object also has a delegate allowing you to customize its behavior.

NSView is an abstract class for all objects displayed in a window. All subclasses implement a drawing method using graphics functions; `drawRect:` (page 3170) is the primary method you override when creating a new `NSView` subclass.

Panels

The `NSPanel` class is a subclass of `NSWindow` that you use to display transient, global, or pressing information. For example, you would use an instance of `NSPanel`, rather than an instance of `NSWindow`, to display error messages or to query the user for a response to remarkable or unusual circumstances. The Application Kit implements some common panels for you such as the Save, Open and Print panels, used to save, open, and print documents. Using these panels gives the user a consistent "look and feel" across applications for common operations.

Menus and Cursors

The `NSMenu`, `NSMenuItem`, and `NSCursor` classes define the look and behavior of the menus and cursors that your application displays to the user.

Grouping and Scrolling Views

The NSBox, NSScrollView, and NSSplitView classes provide graphic "accessories" to other view objects or collections of views in windows. With the NSBox class, you can group elements in windows and draw a border around the entire group. The NSSplitView class lets you "stack" views vertically or horizontally, apportioning to each view some amount of a common territory; a sliding control bar lets the user redistribute the territory among views. The NSScrollView class and its helper class, NSClipView, provide a scrolling mechanism as well as the graphic objects that let the user initiate and control a scroll. The NSRulerView class allows you to add a ruler and markers to a scroll view.

Controlling an Application

The NSControl and NSCell classes, and their subclasses, define a common set of user interface objects such as buttons, sliders, and browsers that the user can manipulate graphically to control some aspect of your application. Just what a particular control affects is up to you: When a control is "touched," it sends an action message to a target object. You typically use Interface Builder to set these targets and actions by Control-dragging from the control object to your application or other object. You can also set targets and actions programmatically.

An NSControl object is associated with one or more NSCell objects that implement the details of drawing and handling events. For example, a button comprises both an NSButton object and an NSButtonCell object. The reason for this separation of functionality is primarily to allow NSCell classes to be reused by NSControl classes. For example, NSMatrix and NSTableView can contain multiple NSCell objects of different types.

Tables

The NSTableView class displays data in row and column form. NSTableView is ideal for, but not limited to, displaying database records, where rows correspond to each record and columns contain record attributes. The user can edit individual cells and rearrange the columns. You control the behavior and content of an NSTableView object by setting its delegate and data source objects.

Text and Fonts

The NSTextField class implements a simple editable text field, and the NSTextView class provides more comprehensive editing features for larger text bodies.

NSTextView, a subclass of the abstract NSText class, defines the interface to Cocoa's extended text system. NSTextView supports rich text, attachments (graphics, file, and other), input management and key binding, and marked text attributes. NSTextView works with the font panel and menu, rulers and paragraph styles, the Services facility (for example, the spell-checking service), and the pasteboard. NSTextView also allows customizing through delegation and notifications—you rarely need to subclass NSTextView. You rarely create instances of NSTextView programmatically either, since objects on Interface Builder's palettes, such as NSTextField, NSForm, and NSScrollView, already contain NSTextView objects.

It is also possible to do more powerful and more creative text manipulation (such as displaying text in a circle) using NSTextStorage, NSLayoutManager, NSTextContainer, and related classes.

The NSFont and NSFontManager classes encapsulate and manage font families, sizes, and variations. The NSFont class defines a single object for each distinct font; for efficiency, these objects, which can be rather large, are shared by all the objects in your application. The NSFontPanel class defines the font specification panel that's presented to the user.

Graphics and Color

The classes NSImage and NSImageRep encapsulate graphics data, allowing you to easily and efficiently access images stored in files on the disk and displayed on the screen. NSImageRep subclasses each know how to draw an image from a particular kind of source data. The presentation of an image is greatly influenced by the hardware that it's displayed on. For example, a particular image may look good on a color monitor, but may be too "rich" for monochrome. Through the image classes, you can group representations of the same image, where each representation fits a specific type of display device—the decision of which representation to use can be left to the NSImage class itself.

Color is supported by the classes NSColor, NSColorPanel, NSColorList, NSColorPicker, and NSColorWell. NSColor supports a rich set of color formats and representations, including custom ones. The other classes are mostly interface classes: They define and present panels and views that allow the user to select and apply colors. For example, the user can drag colors from the color panel to any color well. The NSColorPicking protocol lets you extend the standard color panel.

Dragging

With very little programming on your part, custom view objects can be dragged and dropped anywhere. Objects become part of this dragging mechanism by conforming to NSDragging... protocols: draggable objects conform to the NSDraggingSource protocol, and destination objects (receivers of a drop) conform to the NSDraggingDestination protocol. The Application Kit hides all the details of tracking the cursor and displaying the dragged image.

Printing

The NSPrinter, NSPrintPanel, NSPageLayout, and NSPrintInfo classes work together to provide the means for printing the information that your application displays in its windows and views. You can also create an EPS representation of an NSView.

Accessing the File System

Use the NSFileWrapper class to create objects that correspond to files or directories on disk. NSFileWrapper will hold the contents of the file in memory so that it can be displayed, changed, or transmitted to another application. It also provides an icon for dragging the file or representing it as an attachment. Or use the NSFileManager class in the Foundation framework to access and enumerate file and directory contents. The NSOpenPanel and NSSavePanel classes also provide a convenient and familiar user interface to the file system.

Sharing Data With Other Applications

The NSPasteboard class defines the pasteboard, a repository for data that's copied from your application, making this data available to any application that cares to use it. NSPasteboard implements the familiar cut-copy-paste operation. The NSServicesRequest protocol uses the pasteboard to communicate data that's passed between applications by a registered service.

Checking Spelling

The NSSpellServer class lets you define a spell-checking service and provide it as a service to other applications. To connect your application to a spell-checking service, you use the NSSpellChecker class. The NSIgnoreMisspelledWords and NSChangeSpelling protocols support the spell-checking mechanism.

Localization

If an application is to be used in more than one part of the world, its resources may need to be customized, or "localized," for language, country, or cultural region. For example, an application may need to have separate Japanese, English, French, and German versions of character strings, icons, nib files, or context help. Resource files specific to a particular language are grouped together in a subdirectory of the bundle directory (the directories with the "`.lproj`" extension). Usually you set up localization resource files using Interface Builder. See the specifications for *NSBundle Additions Reference* and `NSBundle` class for more information on localization (`NSBundle` is in the Foundation framework).

Classes

CIColor Additions Reference

Inherits from	NSObject
Conforms to	NSCoding (CIColor) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSColor.h
Availability	Available in Mac OS X v10.4 and later.

Overview

The Application Kit extends the Core Image framework's CIColor class by adding the ability to create an instance of `CIColor` from an existing `NSColor` instance.

Tasks

Creating a CIColor Instance

`- initWithColor:` (page 55)
Initializes a newly allocated `CIColor` object using an `NSColor` object.

Instance Methods

initWithColor:

Initializes a newly allocated `CIColor` object using an `NSColor` object.

```
- (id)initWithColor:(NSColor *)color
```

Parameters

color
The initial color value, which can belong to any available colorspace.

Return Value
The resulting `CIColor` object, or `nil` if the object cannot be initialized with the specified value.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSColor.h`

CIImage Additions Reference

Inherits from	NSObject
Conforms to	NSCoding (CIImage) NSCopying (CIImage) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSCIImageRep.h
Availability	Available in Mac OS X v10.4 and later.

Overview

The Application Kit adds three methods to the Core Image framework's CIImage class.

Tasks

Initializing

`- initWithBitmapImageRep:` (page 59)
Initializes the receiver, a newly allocated `CIImage` object, with the specified bitmap.

Drawing Images

`- drawAtPoint:fromRect:operation:fraction:` (page 58)
Draws all or part of the image at the specified point in the current coordinate system.

`- drawInRect:fromRect:operation:fraction:` (page 58)
Draws all or part of the image in the specified rectangle in the current coordinate system

Instance Methods

drawAtPoint:fromRect:operation:fraction:

Draws all or part of the image at the specified point in the current coordinate system.

```
- (void)drawAtPoint:(NSPoint)point fromRect:(NSRect)srcRect
    operation:(NSCompositingOperation)op fraction:(CGFloat)delta
```

Parameters

point

The location in the current coordinate system at which to draw the image.

srcRect

The source rectangle specifying the portion of the image you want to draw. The coordinates of this rectangle must be specified using the image's own coordinate system.

op

The compositing operation to use when drawing the image.

delta

The opacity of the image, specified as a value from 0.0 to 1.0. Specifying a value of 0.0 draws the image as fully transparent while a value of 1.0 draws the image as fully opaque. Values greater than 1.0 are interpreted as 1.0.

Discussion

The image content is drawn at its current resolution and is not scaled unless the CTM of the current coordinate system itself contains a scaling factor. The image is otherwise positioned and oriented using the current coordinate system.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSCIImageRep.h`

drawInRect:fromRect:operation:fraction:

Draws all or part of the image in the specified rectangle in the current coordinate system

```
- (void)drawInRect:(NSRect)dstRect fromRect:(NSRect)srcRect
    operation:(NSCompositingOperation)op fraction:(CGFloat)delta
```

Parameters

dstRect

The rectangle in which to draw the image.

srcRect

The source rectangle specifying the portion of the image you want to draw. The coordinates of this rectangle must be specified using the image's own coordinate system.

op

The compositing operation to use when drawing the image.

delta

The opacity of the image, specified as a value from 0.0 to 1.0. Specifying a value of 0.0 draws the image as fully transparent while a value of 1.0 draws the image as fully opaque. Values greater than 1.0 are interpreted as 1.0.

Discussion
If the `srcRect` and `dstRect` rectangles have different sizes, the source portion of the image is scaled to fit the specified destination rectangle. The image is otherwise positioned and oriented using the current coordinate system.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
Reducer
SonogramViewDemo

Declared In
`NSCIImageRep.h`

initWithBitmapImageRep:

Initializes the receiver, a newly allocated `CIImage` object, with the specified bitmap.

```
- (id)initWithBitmapImageRep:(NSBitmapImageRep *)bitmapImageRep
```

Parameters

bitmapImageRep

An image representation object containing the bitmap data.

Return Value
The resulting `CIImage` object.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
CocoaSlides
FunHouse
Reducer

Declared In
`NSCIImageRep.h`

NSActionCell Class Reference

Inherits from	NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSActionCell.h
Companion guide	Action Messages
Related sample code	ClockControl TrackBall

Overview

An NSActionCell defines an active area inside a control (an instance of NSControl or one of its subclasses).

As an NSControl's active area, an NSActionCell does three things: it usually performs display of text or an icon; it provides the NSControl with a target and an action; and it handles mouse (cursor) tracking by properly highlighting its area and sending action messages to its target based on cursor movement.

Tasks

Configuring an NSActionCell Object

- `setAlignment:` (page 65)
  Sets the alignment of text in the receiver.
- `setEnabled:` (page 67)
  Sets whether the receiver is enabled or disabled.
- `setFloatingPointFormat:left:right:` (page 67)
  Sets the receiver's floating-point format.
- `setImage:` (page 69)
  Sets the image to be displayed in the receiver.

- `setBezeled:` (page 66) Available in Mac OS X v10.0 through Mac OS X v10.5
  Sets whether the receiver draws itself with a bezeled border.
- `setBordered:` (page 66) Available in Mac OS X v10.0 through Mac OS X v10.5
  Sets whether the receiver draws itself outlined with a plain border.
- `setFont:` (page 68) Available in Mac OS X v10.0 through Mac OS X v10.5
  Sets the font to be used when the receiver displays text.

Obtaining and Setting Cell Values

- `doubleValue` (page 63)
  Returns the receiver's value as a `double` after validating any editing of cell content.
- `integerValue` (page 64)
  Returns the receiver's value as a 64-bit compatible integer after validating any editing of cell content.
- `setObjectValue:` (page 69)
  Discards any editing of the receiver's text and sets its object value to *`object`*.
- `floatValue` (page 64) Available in Mac OS X v10.0 through Mac OS X v10.5
  Returns the receiver's value as a `float` after validating any editing of cell content.
- `intValue` (page 65) Available in Mac OS X v10.0 through Mac OS X v10.5
  Returns the receiver's value as an `int` after validating any editing of cell content.
- `stringValue` (page 70) Available in Mac OS X v10.0 through Mac OS X v10.5
  Returns the receiver's value as a string object as converted by the cell's formatter, if one exists.

Managing the Cell's View

- `setControlView:` (page 67)
  Sets the receiver's control view, the view in which it is drawn.
- `controlView` (page 63) Available in Mac OS X v10.0 through Mac OS X v10.5
  Returns the view in which the receiver was last drawn.

Assigning the Target and Action

- `setAction:` (page 65)
  Sets the selector used for action messages sent by the receiver's control.
- `action` (page 63)
  Returns the receiver's action-message selector.
- `setTarget:` (page 70)
  Sets the receiver's target object.
- `target` (page 71)
  Returns the receiver's target object.

Assigning a Tag

- `setTag:` (page 69)
  Sets the receiver's tag.
- `tag` (page 71)
  Returns the receiver's tag.

Instance Methods

action

Returns the receiver's action-message selector.

```
- (SEL)action
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAction:` (page 65)
- `setTarget:` (page 70)
- `target` (page 71)

Declared In
`NSActionCell.h`

controlView

Returns the view in which the receiver was last drawn. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (NSView *)controlView
```

Return Value
The returned view is normally an `NSControl` object. The method returns `nil` if the receiver has no control view (usually because it hasn't yet been placed in the view hierarchy).

Availability
Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared In
`NSActionCell.h`

doubleValue

Returns the receiver's value as a `double` after validating any editing of cell content. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (double)doubleValue
```

Discussion
If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.5.

See Also
- `validateEditing` (page 845) (NSControl)

Declared In
`NSActionCell.h`

floatValue

Returns the receiver's value as a `float` after validating any editing of cell content. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (float)floatValue
```

Discussion
If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.5.

See Also
- `validateEditing` (page 845) (NSControl)

Declared In
`NSActionCell.h`

integerValue

Returns the receiver's value as a 64-bit compatible integer after validating any editing of cell content.

```
- (NSInteger)integerValue
```

Return Value
A 64-bit compatible integer value, as defined by the `NSInteger` type.

Discussion
If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSActionCell.h`

intValue

Returns the receiver's value as an `int` after validating any editing of cell content. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (int)intValue
```

Discussion
If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.5.

See Also
- `validateEditing` (page 845) (NSControl)

Declared In
`NSActionCell.h`

setAction:

Sets the selector used for action messages sent by the receiver's control.

```
- (void)setAction:(SEL)aSelector
```

Parameters
aSelector
The selector that identifies the action method to invoke.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `action` (page 63)
- `setTarget:` (page 70)
- `target` (page 71)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSActionCell.h`

setAlignment:

Sets the alignment of text in the receiver. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (void)setAlignment:(NSTextAlignment)mode
```

Parameters

mode

One of five constants that specifies alignment within the cell: `NSLeftTextAlignment`, `NSRightTextAlignment`, `NSCenterTextAlignment`, `NSJustifiedTextAlignment`, and `NSNaturalTextAlignment` (the default alignment for the text).

Discussion

The method marks the receiver as needing redisplay after discarding any editing changes that were being made to cell text.

Availability

Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared In

`NSActionCell.h`

setBezeled:

Sets whether the receiver draws itself with a bezeled border. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (void)setBezeled:(BOOL)flag
```

Parameters

flag

`YES` if the cell is to be drawn with a bezeled border, `NO` otherwise.

Discussion

After setting the attribute the method marks the receiver as needing redisplay. The `setBezeled:` and `setBordered:` (page 66) methods are mutually exclusive—that is, a border can be only plain or bezeled.

Availability

Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared In

`NSActionCell.h`

setBordered:

Sets whether the receiver draws itself outlined with a plain border. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (void)setBordered:(BOOL)flag
```

Parameters

flag

`YES` if the cell is to be drawn with a plain border, `NO` otherwise.

Discussion

After setting the attribute the method marks the receiver as needing redisplay. The `setBezeled:` (page 66) and `setBordered:` methods are mutually exclusive—that is, a border can be only plain or bezeled.

Availability

Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared In
NSActionCell.h

setControlView:

Sets the receiver's control view, the view in which it is drawn.

```
- (void)setControlView:(NSView *)view
```

Parameters

view

The view object, which is normally an NSControl *view*. Pass in `nil` if the receiver has no control view (usually because it hasn't yet been placed in the view hierarchy).

Discussion
The control view is typically set in the receiver's implementation of `drawWithFrame:inView:` (page 554) (`NSCell`).

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSActionCell.h

setEnabled:

Sets whether the receiver is enabled or disabled. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (void)setEnabled:(BOOL)flag
```

Parameters

flag

`YES` if the cell is to be enabled, `NO` otherwise

Discussion
The text of disabled cells is changed to gray. If a cell is disabled, it cannot be highlighted, cannot be edited, and does not support mouse tracking (and thus cannot participate in target-action behavior). The method marks the receiver as needing redisplay after discarding any editing changes that were being made to cell text.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared In
NSActionCell.h

setFloatingPointFormat:left:right:

Sets the receiver's floating-point format. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (void)setFloatingPointFormat:(BOOL)autoRange left:(NSUInteger)leftDigits
    right:(NSUInteger)rightDigits
```

Parameters

autoRange

NO if you want the receiver to places digits to the right and left of the decimal point as specified (in leftDigits and rightDigits; YES if you want it to place the digits flexibly.

leftDigits

The maximum number of digits to the left of the decimal point. The receiver might interpret this value flexibly if autoRange is YES.

rightDigits

The maximum number of digits to the right of the decimal point. The receiver might interpret this value flexibly if autoRange is YES.

Discussion

The implementation of this method is based on the NSCell method setFloatingPointFormat:left:right: (page 586). See the description of that method for details.

The NSActionCell implementation of the method supplements the NSCell implementation by marking the receiver as needing redisplay after discarding any editing changes that were being made to cell text.

Note: This method is being deprecated in favor of NSFormatter objects. For more information, see NSFormatter. This documentation is provided only for developers who need to modify older applications.

Availability

Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared In

NSActionCell.h

setFont:

Sets the font to be used when the receiver displays text. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (void)setFont:(NSFont *)fontObj
```

Parameters

fontObj

The font object encapsulating information about the new font. If *fontObj* is nil and the receiver is a text-type cell, the font object currently held by the receiver is autoreleased.

Discussion

If the receiver is not a text-type cell, the method converts it to that type. NSActionCell supplements the NSCell implementation of this method by marking the updated cell as needing redisplay. If the receiver was converted to a text-type cell and is selected, it also updates the field editor with *fontObj*.

Availability

Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared In

NSActionCell.h

setImage:

Sets the image to be displayed in the receiver. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (void)setImage:(NSImage *)image
```

Parameters

image

The image for the receiver to display. If *image* is `nil`, the image currently displayed by the receiver is removed.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared In
`NSActionCell.h`

setObjectValue:

Discards any editing of the receiver's text and sets its object value to *object*. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (void)setObjectValue:(id < NSCopying >)object
```

Parameters

object

The object value to assign to the receiver.

Discussion
If the object value is afterward different from what it was before the method was invoked, the method marks the receiver as needing redisplay.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared In
`NSActionCell.h`

setTag:

Sets the receiver's tag.

```
- (void)setTag:(NSInteger)anInt
```

Parameters

anInt

An integer tag to be associated with the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `tag` (page 71)

Declared In
NSActionCell.h

setTarget:

Sets the receiver's target object.

```
- (void)setTarget:(id)anObject
```

Parameters

anObject
The object that is the target of action messages sent by the receiver's control.

Availability
Available in Mac OS X v10.0 and later.

See Also
- action (page 63)
- setAction: (page 65)
- target (page 71)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSActionCell.h

stringValue

Returns the receiver's value as a string object as converted by the cell's formatter, if one exists. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (NSString *)stringValue
```

Discussion
If no formatter exists and the value is an NSString, returns the value as a plain, attributed, or localized formatted string. If the value is not an NSString or cannot be converted to one, returns an empty string. The method supplements the NSCell implementation by validating and retaining any editing changes being made to cell text.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.5.

See Also
- validateEditing (page 845) (NSControl)

Declared In
NSActionCell.h

tag

Returns the receiver's tag.

```
- (NSInteger)tag
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTag:` (page 69)

Declared In
`NSActionCell.h`

target

Returns the receiver's target object.

```
- (id)target
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `action` (page 63)
- `setAction:` (page 65)
- `setTarget:` (page 70)

Related Sample Code
ClockControl

Declared In
`NSActionCell.h`

NSAffineTransform Additions Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSAffineTransform.h
Companion guide	Cocoa Drawing Guide

Overview

The Application Kit extends Foundation's NSAffineTransform class by adding:

- Methods for applying affine transformations to the current graphics context.
- A method for applying an affine transformation to an `NSBezierPath`.

Note: In Mac OS X v10.3 and earlier the `NSAffineTransform` class was declared and implemented entirely in the Application Kit framework. As of Mac OS X v10.4 the `NSAffineTransform` class has been split across the Foundation Kit and Application Kit frameworks.

Tasks

Setting and Building the Current Transformation Matrix

- `set` (page 74)
  Sets the current transformation matrix to the receiver's transformation matrix.
- `concat` (page 74)
  Appends the receiver's matrix to the current transformation matrix stored in the current graphics context, replacing the current transformation matrix with the result.

Transforming Bezier Paths

- `transformBezierPath:` (page 75)
  Creates and returns a new `NSBezierPath` object with each point in the given path transformed by the receiver.

Instance Methods

concat

Appends the receiver's matrix to the current transformation matrix stored in the current graphics context, replacing the current transformation matrix with the result.

```
- (void)concat
```

Discussion
Concatenation is performed by matrix multiplication—see "Manipulating Transform Values".

If this method is invoked from within an `NSView` `drawRect:` (page 3170) method, then the current transformation matrix is an accumulation of the screen, window, and any superview's transformation matrices. Invoking this method defines a new user coordinate system whose coordinates are mapped into the former coordinate system according to the receiver's transformation matrix. To undo the concatenation, you must invert the receiver's matrix and invoke this method again.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `set` (page 74)
- `invert`

Related Sample Code
DockTile
Sketch+Accessibility
Sketch-112
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSAffineTransform.h`

set

Sets the current transformation matrix to the receiver's transformation matrix.

```
- (void)set
```

Discussion
The current transformation is stored in the current graphics context and is applied to subsequent drawing operations. You should use this method sparingly because it removes the existing transformation matrix, which is an accumulation of transformation matrices for the screen, window, and any superviews. Instead use the `concat` (page 74) method to add this transformation matrix to the current transformation matrix.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
FunHouse

Declared In
`NSAffineTransform.h`

transformBezierPath:

Creates and returns a new `NSBezierPath` object with each point in the given path transformed by the receiver.

```
- (NSBezierPath *)transformBezierPath:(NSBezierPath *)aPath
```

Parameters
aPath
An object representing the bezier path to be used in the transformation.

Discussion
The original `NSBezierPath` object is not modified.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- transformPoint:transformSize:`

Related Sample Code
Cropped Image
Polygons

Declared In
`NSAffineTransform.h`

NSAlert Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSAlert.h
Availability	Available in Mac OS X v10.3 and later.
Companion guides	Dialogs and Special Panels Sheet Programming Topics
Related sample code	ExtractMovieAudioToAIFF QTExtractAndConvertToAIFF QTExtractAndConvertToMovieFile QTKitTimeCode QTRecorder

Overview

You use an `NSAlert` object to display an alert, either as an application-modal dialog or as a sheet attached to a document window. The methods of the `NSAlert` class allow you to specify alert level, icon, button titles, and alert text. The class also lets your alerts display help buttons and provides ways for applications to offer help specific to an alert. To display an alert as a sheet, invoke the `beginSheetModalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 84) method; to display one as an application-modal dialog, use the `runModal` (page 88) method.

By design, an `NSAlert` object is intended for a single alert—that is, an alert with a unique combination of title, buttons, and so on—that is displayed upon a particular condition. You should create an `NSAlert` object for each alert dialog. Normally you should create an `NSAlert` object when you need to display an alert, and release it when you are done. If you have a particular alert dialog that you need to show repeatedly, you can retain and reuse an instance of `NSAlert` for this dialog.

After creating an alert using one of the alert creation methods, you can customize it further prior to displaying it by customizing its attributes. See "Instance Attributes" (page 78)

Note: The `NSAlert` class, which was introduced in Mac OS X v10.3, supersedes the functional Application Kit API for displaying alerts (`NSRunAlertPanel`, `NSBeginAlertSheet`, and so on). The former API is still supported, but you should use the `NSAlert` class for your application's alert dialogs.

Instance Attributes

`NSAlert` objects have the following attributes:

- **Type.** An alert's type helps convey the importance or gravity of its message to the user. Specified with `setAlertStyle:` (page 89).
- **Message text.** The main message of the alert. Specified with `setMessageText:` (page 92).
- **Informative text.** Additional information about the alert. Specified with `informativeText` (page 86).
- **icon.** The icon displayed in the alert. Specified with : `setIcon:` (page 91).
- **Help.** Alerts can let the user get help about them. Use `setHelpAnchor:` (page 90) and `setShowsHelp:` (page 92).
- **Response buttons.** By default an alert has one response button: the OK button. You can add more response buttons using: `addButtonWithTitle:` (page 83).
- **Suppression checkbox.** A suppression checkbox allows the user to suppress the display of a particular alert in subsequent occurrences of the event that triggers it. Use `setShowsSuppressionButton:` (page 93), `suppressionButton` (page 95).
- **Accessory view.** An accessory view lets you add additional information to an alert; for example, a text field with contact information. Use `setAccessoryView:` (page 88), `layout` (page 87).

Subclassing Notes

The `NSAlert` class is not designed for subclassing.

Tasks

Creating Alerts

`+ alertWithError:` (page 80)

Returns an alert initialized from information in an error object.

`+ alertWithMessageText:defaultButton:alternateButton:otherButton:informativeTextWithFormat:` (page 81)

Creates an alert compatible with alerts created using the `NSRunAlertPanel` (page 3998) function for display as a warning-style alert.

Configuring Alerts

- `layout` (page 87)
  Specifies that the receiver must do immediate layout instead of lazily just before display.
- `alertStyle` (page 83)
  Returns the `NSAlertStyle` constant identifying the receiver's alert style.
- `setAlertStyle:` (page 89)
  Sets the alert style of the receiver.
- `accessoryView` (page 82)
  Returns the receiver's accessory view.
- `setAccessoryView:` (page 88)
  Sets the receiver's accessory view.
- `showsHelp` (page 94)
  Indicates whether the receiver has a help button.
- `setShowsHelp:` (page 92)
  Specifies whether the receiver has a help button.
- `helpAnchor` (page 86)
  Returns the receiver's HTML help anchor.
- `setHelpAnchor:` (page 90)
  Associates the receiver to a given anchor.
- `delegate` (page 85)
  Returns the receiver's delegate.
- `setDelegate:` (page 90)
  Sets the receiver's delegate.

Displaying Alerts

- `runModal` (page 88)
  Runs the receiver as an application-modal dialog and returns the constant positionally identifying the button clicked.
- `beginSheetModalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 84)
  Runs the receiver modally as an alert sheet attached to a specified window.
- `suppressionButton` (page 95)
  Returns the receiver's suppression checkbox.
- `showsSuppressionButton` (page 94)
  Indicates whether the receiver shows a suppression button.
- `setShowsSuppressionButton:` (page 93)
  Specifies whether the receiver includes a suppression checkbox.

Accessing Alert Text

- `informativeText` (page 86)
  Returns the receiver's informative text.

- `setInformativeText:` (page 91)
  Sets the receiver's informative text to a given text.
- `messageText` (page 87)
  Returns the receiver's message text (or title).
- `setMessageText:` (page 92)
  Sets the receiver's message text, or title, to a given text.

Accessing Alert Icons

- `icon` (page 86)
  Returns the icon displayed in the receiver.
- `setIcon:` (page 91)
  Sets the icon to be displayed in the alert to a given icon.

Accessing Alert Buttons

- `buttons` (page 85)
  Returns the receiver's buttons.
- `addButtonWithTitle:` (page 83)
  Adds a button with a given title to the receiver.

Getting Alert Panels

- `window` (page 95)
  Provides the application-modal panel associated with the receiver.

Class Methods

alertWithError:

Returns an alert initialized from information in an error object.

```
+ (NSAlert *)alertWithError:(NSError *)error
```

Parameters

error
  Error information to display.

Return Value
Initialized alert.

Discussion
The `NSAlert` class extracts the localized error description, recovery suggestion, and recovery options from *error* and uses them as the alert's message text, informative text, and button titles, respectively.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
AudioDataOutputToAudioUnit
QTRecorder
Quartz Composer WWDC 2005 TextEdit
SimpleStickies
StillMotion

Declared In
NSAlert.h

alertWithMessageText:defaultButton:alternateButton:otherButton: informativeTextWithFormat:

Creates an alert compatible with alerts created using the NSRunAlertPanel (page 3998) function for display as a warning-style alert.

```
+ (NSAlert *)alertWithMessageText:(NSString *)messageTitle defaultButton:(NSString
    *)defaultButtonTitle alternateButton:(NSString *)alternateButtonTitle
   otherButton:(NSString *)otherButtonTitle informativeTextWithFormat:(NSString
   *)informativeText, ...
```

Parameters

messageTitle
Title of the alert. When nil or an empty string, a default localized title is used (“Alert” in English).

defaultButtonTitle
Title for the default button. When nil or an empty string, a default localized button title (“OK” in English) is used.

alternateButtonTitle
Title for the alternate button. When nil, the alternate button is not created.

otherButtonTitle
Title for the other button. When nil, the other button is not created.

informativeText
Informative text. This is optional but must be an empty string (@"") not nil. Can embed variable values using a format string; list any necessary arguments for this formatted string at the end of the method’s argument list. For more information on format strings, see Formatting String Objects.

Return Value
Initialized alert.

Discussion
For languages that read left to right, the buttons are laid out on the bottom-right corner of the alert sheet or window, with *defaultButtonTitle* on the right, *alternateButtonTitle* on the left, and *otherButtonTitle* in the middle. The return values identifying these buttons are constants—NSAlertDefaultReturn, NSAlertAlternateReturn, and NSAlertOtherReturn—that correspond to the keywords.

By default, the first button has a key equivalent of Return, any button with a title of "Cancel" has a key equivalent of Escape, and any button with the title "Don't Save" has a key equivalent of Command-D (but only if it is not the first button). You can also assign different key equivalents for the buttons using the `setKeyEquivalent:` (page 483) method of the `NSButton` class. To access the alert's buttons, use the `buttons` (page 85) method.

Special Considerations

This is a compatibility method. It is designed for easy adoption by applications migrating from the corresponding function-based API. This method uses earlier return values—`NSAlertDefaultReturn`, `NSAlertAlternateReturn`, and `NSAlertOtherReturn`—compatible with the earlier API, rather than the return values defined by the `NSAlert` class, described in "Button Return Values" (page 96).

Unless you must maintain compatibility with existing alert-processing code that uses the function-based API, you should allocate (`alloc`) and initialize (`init`) the object, and then set its attributes using the appropriate methods of the `NSAlert` class. However, using `init` results in no default buttons being provided, in which case you must call `addButtonWithTitle:` (page 83) explicitly.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
Denoise
ExtractMovieAudioToAIFF
QTExtractAndConvertToAIFF
QTExtractAndConvertToMovieFile
QTKitTimeCode

Declared In
`NSAlert.h`

Instance Methods

accessoryView

Returns the receiver's accessory view.

```
- (NSView *)accessoryView
```

Return Value
The alert's accessory view.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setAccessoryView:` (page 88)

Declared In
`NSAlert.h`

addButtonWithTitle:

Adds a button with a given title to the receiver.

```
- (NSButton *)addButtonWithTitle:(NSString *)buttonTitle
```

Parameters

buttonTitle

Title of the button to add to the alert. Must not be `nil`.

Return Value

Button added to the alert.

Discussion

Buttons are placed starting near the right side of the alert and going toward the left side (for languages that read left to right). The first three buttons are identified positionally as `NSAlertFirstButtonReturn`, `NSAlertSecondButtonReturn`, `NSAlertThirdButtonReturn` in the return-code parameter evaluated by the modal delegate. Subsequent buttons are identified as `NSAlertThirdButtonReturn` +*n*, where *n* is an integer

By default, the first button has a key equivalent of Return, any button with a title of "Cancel" has a key equivalent of Escape, and any button with the title "Don't Save" has a key equivalent of Command-D (but only if it is not the first button). You can also assign different key equivalents for the buttons using the `setKeyEquivalent:` (page 483) method of the `NSButton` class. In addition, you can use the `setTag:` (page 838) method of the `NSButton` class to set the return value.

`NSAlert` instances created with the `init` method lack any buttons. You must call `addButtonWithTitle:` (page 83) to add a minimum of one button.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- buttons` (page 85)

Related Sample Code

CoreRecipes

IdentitySample

UIElementInspector

Declared In

`NSAlert.h`

alertStyle

Returns the `NSAlertStyle` constant identifying the receiver's alert style.

```
- (NSAlertStyle)alertStyle
```

Return Value

Alert style for the alert. See NSAlertStyle (page 96) for the list of alert style constants.

Availability

Available in Mac OS X v10.3 and later.

See Also
- `setAlertStyle:` (page 89)

Declared In
`NSAlert.h`

beginSheetModalForWindow:modalDelegate:didEndSelector:contextInfo:

Runs the receiver modally as an alert sheet attached to a specified window.

```
- (void)beginSheetModalForWindow:(NSWindow *)window modalDelegate:(id)modalDelegate
    didEndSelector:(SEL)alertDidEndSelector contextInfo:(void *)contextInfo
```

Parameters

window
The parent window for the sheet.

modalDelegate
The delegate for the modal-dialog session.

alertDidEndSelector
Message the alert sends to *modalDelegate* after the user responds but before the sheet is dismissed.

contextInfo
Contextual data passed to *modalDelegate* in *didEndSelector* message.

Discussion
You can create the required `NSAlert` object either through the standard allocate-initialize procedure or by using the compatibility method `alertWithMessageText:defaultButton:alternateButton:otherButton:informativeTextWithFormat:` (page 81).

The *alertDidEndSelector* argument must be a selector that takes three arguments, and the corresponding method should have a declaration modeled on the following example:

```
- (void) alertDidEnd:(NSAlert *)alert returnCode:(NSInteger)returnCode
contextInfo:(void *)contextInfo;
```

where *alert* is the `NSAlert` object, *returnCode* specifies which button the user pressed, and *contextInfo* is the same *contextInfo* passed in the original message. The *returnCode* argument identifies which button was used to dismiss the alert (see this method's "Special Considerations" section). The modal delegate determines which button was clicked ("OK", "Cancel", and so on) and proceeds accordingly.

If you want to dismiss the sheet from within the *alertDidEndSelector* method before the modal delegate carries out an action in response to the return value, send `orderOut:` (page 3352) (`NSWindow`) to the window object obtained by sending `window` (page 95) to the *alert* argument. This allows you to chain sheets, for example, by dismissing one sheet before showing the next from within the *alertDidEndSelector* method. Note that you should be careful not to call `orderOut:` on the sheet from elsewhere in your program before the *alertDidEndSelector* method is invoked.

Special Considerations

When you use `alertWithMessageText:defaultButton:alternateButton:otherButton:informativeTextWithFormat:` (page 81) to create an alert, these are the constants used to identify the button used to dismiss the alert: `NSAlertDefaultReturn`, `NSAlertAlternateReturn`, and `NSAlertOtherReturn`. Otherwise, the constants used are the ones described in "Button Return Values" (page 96).

Availability
Available in Mac OS X v10.3 and later.

See Also
`- runModal` (page 88)

Related Sample Code
Cocoa Tips and Tricks
CoreRecipes
IdentitySample
QTRecorder
XMLBrowser

Declared In
`NSAlert.h`

buttons

Returns the receiver's buttons.

```
- (NSArray *)buttons
```

Return Value
The alert's buttons. The rightmost button is at index 0.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- addButtonWithTitle:` (page 83)

Declared In
`NSAlert.h`

delegate

Returns the receiver's delegate.

```
- (id < NSAlertDelegate >)delegate
```

Return Value
The alert's delegate.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setDelegate:` (page 90)

Declared In
`NSAlert.h`

helpAnchor

Returns the receiver's HTML help anchor.

```
- (NSString *)helpAnchor
```

Return Value
The alert's help anchor. It's `nil` when the alert has no help anchor.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setHelpAnchor:` (page 90)

Declared In
`NSAlert.h`

icon

Returns the icon displayed in the receiver.

```
- (NSImage *)icon
```

Return Value
The alert's icon.

Discussion
The default image is the application icon (`NSApplicationIcon` application property).

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setIcon:` (page 91)

Declared In
`NSAlert.h`

informativeText

Returns the receiver's informative text.

```
- (NSString *)informativeText
```

Return Value
The alert's informative text.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setInformativeText:` (page 91)
- `messageText` (page 87)

Declared In
`NSAlert.h`

layout

Specifies that the receiver must do immediate layout instead of lazily just before display.

```
- (void)layout
```

Discussion
You need to call this method only when you need to customize the alert's layout. Call this method after all the alert's attributes have been customized, including the suppression checkbox and the accessory layout. After the method returns, you can make the necessary layout changes; for example, adjusting the frame of the accessory view.

Note: The standard alert layout is subject to change in future system software versions. Therefore, if you rely on custom alert layout, you should make sure your layouts work as expected in future releases of Mac OS.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setAccessoryView:` (page 88)

Declared In
`NSAlert.h`

messageText

Returns the receiver's message text (or title).

```
- (NSString *)messageText
```

Return Value
The alert's message text.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setMessageText:` (page 92)

- informativeText (page 86)

Declared In
NSAlert.h

runModal

Runs the receiver as an application-modal dialog and returns the constant positionally identifying the button clicked.

```
- (NSInteger)runModal
```

Return Value
Response to the alert. See this method's "Special Considerations" section for details.

Discussion
You can create the alert either through the standard allocate–initialize procedure or by using the compatibility method alertWithMessageText:defaultButton:alternateButton:otherButton:informativeTextWithFormat: (page 81).

Special Considerations

When you use alertWithMessageText:defaultButton:alternateButton:otherButton:informativeTextWithFormat: (page 81) to create an alert, these are the constants used to identify the button used to dismiss the alert: NSAlertDefaultReturn, NSAlertAlternateReturn, and NSAlertOtherReturn. Otherwise, the constants used are the ones described in "Button Return Values" (page 96).

Availability
Available in Mac OS X v10.3 and later.

See Also
- beginSheetModalForWindow:modalDelegate:didEndSelector:contextInfo: (page 84)

Related Sample Code
AudioDataOutputToAudioUnit
Denoise
QTKitTimeCode
Quartz Composer WWDC 2005 TextEdit
StillMotion

Declared In
NSAlert.h

setAccessoryView:

Sets the receiver's accessory view.

```
- (void)setAccessoryView:(NSView *)accessoryView
```

Parameters
accessoryView
View that is to be the alert's accessory view.

Discussion

The `NSAlert` class places the accessory view between the informative text or suppression checkbox (if present) and the response buttons. To change the location of the accessory view, you must first call the `layout` (page 87) method.

Listing 5-1 shows an example of adding an accessory view to an alert. Figure 5-1 shows the alert generated.

Listing 5-1 Adding an accessory view to an alert

```
NSTextView *accessory = [[NSTextView alloc] initWithFrame:NSMakeRect(0,0,200,15)];
NSFont *font = [NSFont systemFontOfSize:[NSFont systemFontSize]];
NSDictionary *textAttributes = [NSDictionary dictionaryWithObject:font
forKey:NSFontAttributeName];
[accessory insertText:[[NSAttributedString alloc] initWithString:@"Text in
accessory view"
                                                        attributes:textAttributes]];
[accessory setEditable:NO];
[accessory setDrawsBackground:NO];

NSAlert* alert = [NSAlert alertWithMessageText:@"Message text" defaultButton:nil
 alternateButton:nil otherButton:nil informativeTextWithFormat:@"Informative
text"];
[alert setAccessoryView:accessory];
[alert runModal];
```

Figure 5-1 Alert dialog with an accessory view

Availability

Available in Mac OS X v10.5 and later.

See Also

`- accessoryView` (page 82)

Declared In

`NSAlert.h`

setAlertStyle:

Sets the alert style of the receiver.

```
- (void)setAlertStyle:(NSAlertStyle)style
```

Parameters

style

Alert style for the alert. Indicates the severity level of the alert. See NSAlertStyle (page 96) for the list of alert style constants.

Availability
Available in Mac OS X v10.3 and later.

See Also
- alertStyle (page 83)

Related Sample Code
CocoaDVDPlayer
CoreRecipes
IdentitySample
UIElementInspector

Declared In
NSAlert.h

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSAlertDelegate >)delegate
```

Parameters

delegate

Delegate for the alert. nil removes the delegate.

Availability
Available in Mac OS X v10.3 and later.

See Also
- delegate (page 85)

Declared In
NSAlert.h

setHelpAnchor:

Associates the receiver to a given anchor.

```
- (void)setHelpAnchor:(NSString *)anchor
```

Parameters

anchor

Anchor to associate with the alert. nil removes the associated help anchor.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `helpAnchor` (page 86),
- `setShowsHelp:` (page 92)

Declared In
`NSAlert.h`

setIcon:

Sets the icon to be displayed in the alert to a given icon.

```
- (void)setIcon:(NSImage *)icon
```

Parameters

icon

Icon for the alert. `nil` restores the application icon.

Discussion
By default, the image is the application icon, accessed via the application bundle's `NSApplicationIcon` property.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `icon` (page 86)

Declared In
`NSAlert.h`

setInformativeText:

Sets the receiver's informative text to a given text.

```
- (void)setInformativeText:(NSString *)informativeText
```

Parameters

informativeText

Informative text for the alert. The value must not be `nil`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `informativeText` (page 86)
- `setMessageText:` (page 92)

Related Sample Code
CocoaDVDPlayer
CoreRecipes
IdentitySample
UIElementInspector

XMLBrowser

Declared In
NSAlert.h

setMessageText:

Sets the receiver's message text, or title, to a given text.

```
- (void)setMessageText:(NSString *)messageText
```

Parameters

messageText
Message text for the alert.

Availability
Available in Mac OS X v10.3 and later.

See Also
- messageText (page 87)
- setInformativeText: (page 91)

Related Sample Code
CocoaDVDPlayer
CoreRecipes
IdentitySample
UIElementInspector
XMLBrowser

Declared In
NSAlert.h

setShowsHelp:

Specifies whether the receiver has a help button.

```
- (void)setShowsHelp:(BOOL)showsHelp
```

Parameters

showsHelp
YES for a help button, NO for no help button.

Discussion
When the help button is pressed, the alert delegate (delegate (page 85)) is first sent a alertShowHelp: (page 3551) message. If there is no delegate, or the delegate does not implement alertShowHelp: (page 3551) or returns NO, then the openHelpAnchor:inBook: (page 1318) message is sent to the application's help manager with a nil book and the anchor specified by setHelpAnchor: (page 90), if any. An exception is raised if the delegate returns NO and no help anchor is set.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setDelegate:` (page 90)
- `showsHelp` (page 94)

Declared In
`NSAlert.h`

setShowsSuppressionButton:

Specifies whether the receiver includes a suppression checkbox.

```
- (void)setShowsSuppressionButton:(BOOL)showButton
```

Parameters

showButton

When `YES` the alert includes the suppression checkbox.

Discussion

You can set the title of the checkbox with the following code:

```
[[alert suppressionButton] setTitle:title];
```

Listing 5-2 shows how to add a suppression checkbox (with the default suppression-checkbox title) to a modal alert. Figure 5-2 shows the corresponding dialog.

Listing 5-2 Creating an alert with a suppression checkbox

```
NSString *exampleAlertSuppress = @"ExampleAlertSuppress";
NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
if ([defaults boolForKey:exampleAlertSuppress]) {
    NSLog(@"ExampleAlert suppressed");
}
else {
    NSAlert* alert = [NSAlert alertWithMessageText:@"Message text"
defaultButton:nil alternateButton:nil
                                        otherButton:nil
informativeTextWithFormat:@"Informative text"];
    [alert setShowsSuppressionButton:YES];
    [alert runModal];
    if ([[alert suppressionButton] state] == NSOnState) {
        // Suppress this alert from now on.
        [defaults setBool:YES forKey:exampleAlertSuppress];
    }
    [alert release];
}
```

Figure 5-2 Alert dialog with a suppression checkbox

Availability
Available in Mac OS X v10.5 and later.

See Also
- `suppressionButton` (page 95)

Declared In
`NSAlert.h`

showsHelp

Indicates whether the receiver has a help button.

```
- (BOOL)showsHelp
```

Return Value
`YES` if the alert has a help button, `NO` otherwise.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setShowsHelp:` (page 92)

Declared In
`NSAlert.h`

showsSuppressionButton

Indicates whether the receiver shows a suppression button.

```
- (BOOL)showsSuppressionButton
```

Return Value
`YES` when the alert shows a suppression button, `NO` otherwise. The default is `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setShowsSuppressionButton: (page 93)

Declared In
NSAlert.h

suppressionButton

Returns the receiver's suppression checkbox.

```
- (NSButton *)suppressionButton
```

Return Value
The alert's suppression button.

Discussion
You can use this method to customize the alert's suppression checkbox before the alert is displayed. For example, you can change the title of the checkbox or specify its initial state, which is unselected by default.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSAlert.h

window

Provides the application-modal panel associated with the receiver.

```
- (id)window
```

Return Value
The receiver's associated NSPanel object.

Discussion
This method is useful when you want to dismiss an alert created with beginSheetModalForWindow:modalDelegate:didEndSelector:contextInfo: (page 84) within the method identified by the didEndSelector: parameter.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
BackgroundExporter
QTKitTimeCode

Declared In
NSAlert.h

Constants

NSAlertStyle

The `NSAlert` class defines the alert styles used by `setAlertStyle:` (page 89) and `alertStyle` (page 83).

```
enum {
    NSWarningAlertStyle = 0,
    NSInformationalAlertStyle = 1,
    NSCriticalAlertStyle = 2
};
typedef NSUInteger NSAlertStyle;
```

Constants

`NSWarningAlertStyle`

An alert used to warn the user about a current or impending event. The purpose is more than informational but not critical. This is the default alert style.

Available in Mac OS X v10.3 and later.

Declared in `NSAlert.h`.

`NSInformationalAlertStyle`

An alert used to inform the user about a current or impending event.

Available in Mac OS X v10.3 and later.

Declared in `NSAlert.h`.

`NSCriticalAlertStyle`

Reserved this style for critical alerts, such as when there might be severe consequences as a result of a certain user response (for example, a "clean install" will erase all data on a volume). This style causes the icon to be badged with a caution icon.

Available in Mac OS X v10.3 and later.

Declared in `NSAlert.h`.

Discussion

Currently, there is no visual difference between informational and warning alerts. You should only use the critical (or "caution") alert style if warranted, as specified in the "Alerts" chapter in *Apple Human Interface Guidelines*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`NSAlert.h`

Button Return Values

An alert's return values for buttons are position dependent. The following constants describe the return values for the first three buttons on an alert (assuming a language that reads left to right).

```
enum {
   NSAlertFirstButtonReturn  = 1000,
   NSAlertSecondButtonReturn  = 1001,
   NSAlertThirdButtonReturn  = 1002
};
```

Constants

`NSAlertFirstButtonReturn`

The user clicked the first (rightmost) button on the dialog or sheet.

Available in Mac OS X v10.3 and later.

Declared in `NSAlert.h`.

`NSAlertSecondButtonReturn`

The user clicked the second button from the right edge of the dialog or sheet.

Available in Mac OS X v10.3 and later.

Declared in `NSAlert.h`.

`NSAlertThirdButtonReturn`

The user clicked the third button from the right edge of the dialog or sheet.

Available in Mac OS X v10.3 and later.

Declared in `NSAlert.h`.

Discussion

If you have more than three buttons on your alert, the button-position return value is `NSAlertThirdButtonReturn` + *n*, where *n* is an integer. For languages that read right to left, the first button's position is closest to the left edge of the dialog or sheet.

Declared In

`NSAlert.h`

NSAnimation Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	AppKit/NSAnimation.h
Companion guides	Animation Programming Guide for Cocoa Cocoa Drawing Guide
Related sample code	iSpend Reducer

Overview

Objects of the `NSAnimation` class manage the timing and progress of animations in the user interface. The class also lets you link together multiple animations so that when one animation ends another one starts. It does not provide any drawing support for animation and does not directly deal with views, targets, or actions.

Note: For simple tasks requiring a timing mechanism, consider using `NSTimer`.

`NSAnimation` objects have several characteristics, including duration, frame rate, and animation curve, which describes the relative speed of the animation over its course. You can set progress marks in an animation, each of which specifies a percentage of the animation completed; when an animation reaches a progress mark, it notifies its delegate and posts a notification to any observers. Animations execute in one of three blocking modes: blocking, non-blocking on the main thread, and non-blocking on a separate thread. The non-blocking modes permit the handling of user events while the animation is running.

Subclassing Notes

The usual usage pattern for `NSAnimation` is to make a subclass that overrides (at least) the setCurrentProgress: (page 109) method to invoke the superclass implementation and then perform whatever animation action is needed. The method implementation might invoke currentValue (page 104) and then use that value to update some drawing; as a consequence of invoking currentValue (page 104), the method

`animation:valueForProgress:` (page 3560) is sent to the delegate (if there is a delegate that implements the method). For more information on subclassing `NSAnimation`, see *Animation Programming Guide for Cocoa*.

Tasks

Initializing an NSAnimation Object

- `initWithDuration:animationCurve:` (page 106)
  Returns an `NSAnimation` object initialized with the specified duration and animation-curve values.

Configuring an Animation

- `setAnimationBlockingMode:` (page 108)
  Sets the blocking mode of the receiver.
- `animationBlockingMode` (page 102)
  Returns the blocking mode the receiver is next scheduled to run under.
- `runLoopModesForAnimating` (page 107)
  Overridden to return the run-loop modes that the receiver uses to run the animation timer in.
- `setAnimationCurve:` (page 108)
  Sets the receiver's animation curve.
- `animationCurve` (page 103)
  Returns the animation curve the receiver is running under.
- `setDuration:` (page 110)
  Sets the duration of the animation to a specified number of seconds.
- `duration` (page 105)
  Returns the duration of the animation, in seconds.
- `setFrameRate:` (page 110)
  Sets the frame rate of the receiver.
- `frameRate` (page 105)
  Returns the frame rate of the animation.

Managing the Delegate

- `setDelegate:` (page 109)
  Sets the delegate of the receiver.
- `delegate` (page 105)
  Returns the delegate of the receiver.

Controlling and Monitoring an Animation

- `startAnimation` (page 111)
  Starts the animation represented by the receiver.
- `stopAnimation` (page 112)
  Stops the animation represented by the receiver.
- `isAnimating` (page 106)
  Returns a Boolean value that indicates whether the receiver is currently animating.
- `setCurrentProgress:` (page 109)
  Sets the current progress of the receiver.
- `currentProgress` (page 104)
  Returns the current progress of the receiver.
- `currentValue` (page 104)
  Returns the current value of the effect based on the current progress.

Managing Progress Marks

- `addProgressMark:` (page 102)
  Adds the progress mark to the receiver.
- `removeProgressMark:` (page 107)
  Removes progress mark from the receiver.
- `setProgressMarks:` (page 111)
  Sets the receiver's progress marks to the values specified in the passed-in array.
- `progressMarks` (page 107)
  Returns the receiver's progress marks.

Linking Animations Together

- `startWhenAnimation:reachesProgress:` (page 112)
  Starts running the animation represented by the receiver when another animation reaches a specific progress mark.
- `stopWhenAnimation:reachesProgress:` (page 113)
  Stops running the animation represented by the receiver when another animation reaches a specific progress mark.
- `clearStartAnimation` (page 103)
  Clears linkage to another animation that causes the receiver to start.
- `clearStopAnimation` (page 103)
  Clears linkage to another animation that causes the receiver to stop.

Instance Methods

addProgressMark:

Adds the progress mark to the receiver.

```
- (void)addProgressMark:(NSAnimationProgress)progressMark
```

Parameters

progressMark

A `float` value (typed as `NSAnimationProgress`) between 0.0 and 1.0. Values outside that range are pinned to 0.0 or 1.0, whichever is nearest.

Discussion

A progress mark represents a percentage of the animation completed. When the animation reaches a progress mark, an `animation:didReachProgressMark:` (page 3560) message is sent to the delegate and an `NSAnimationProgressMarkNotification` (page 116) is broadcast to all observers. You might receive multiple notifications of progress advances over multiple marks.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `currentProgress` (page 104)
- `removeProgressMark:` (page 107)

Declared In

`NSAnimation.h`

animationBlockingMode

Returns the blocking mode the receiver is next scheduled to run under.

```
- (NSAnimationBlockingMode)animationBlockingMode
```

Return Value

A constant representing the receiver's blocking mode. See "NSAnimationBlockingMode" (page 114) for valid values.

Discussion

The animation can run in blocking mode or non-blocking mode; non-blocking mode can be either on the main thread or on a separate thread. The default mode is `NSAnimationBlocking`.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `setAnimationBlockingMode:` (page 108)

Declared In

`NSAnimation.h`

animationCurve

Returns the animation curve the receiver is running under.

```
- (NSAnimationCurve)animationCurve
```

Return Value
An `NSAnimationCurve` constant indicating the animation curve.

Discussion
The animation curve describes the relative frame rate over the course of the animation. See "NSAnimationCurve" (page 113) for valid `NSAnimationCurve` constants.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setAnimationCurve:` (page 108)

Declared In
`NSAnimation.h`

clearStartAnimation

Clears linkage to another animation that causes the receiver to start.

```
- (void)clearStartAnimation
```

Discussion
The linkage to the other animation is made with `startWhenAnimation:reachesProgress:` (page 112).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `startAnimation` (page 111)

Declared In
`NSAnimation.h`

clearStopAnimation

Clears linkage to another animation that causes the receiver to stop.

```
- (void)clearStopAnimation
```

Discussion
The linkage to the other animation is made with `stopWhenAnimation:reachesProgress:` (page 113).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `stopAnimation` (page 112)

Declared In
NSAnimation.h

currentProgress

Returns the current progress of the receiver.

```
- (NSAnimationProgress)currentProgress
```

Return Value
A float value typed as NSAnimationProgress that indicates the current progress of the animation.

Discussion
The current progress is a value between 0.0 and 1.0 that represents the percentage of the animation currently completed.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setCurrentProgress: (page 109)

Declared In
NSAnimation.h

currentValue

Returns the current value of the effect based on the current progress.

```
- (float)currentValue
```

Return Value
A float value that indicates the current value of the animation effect.

Discussion
NSAnimation gets the current value from the delegate in animation:valueForProgress: (page 3560) or, if that method is not implemented, computes it from the current progress by factoring in the animation curve. NSAnimation itself does not invoke this method currently. Instances of NSAnimation subclasses or other objects can invoke this method on a periodic basis to get the current value.

Although this method has no corresponding setter method, those NSAnimation subclasses may override this method to return a custom curve value instead of implementing animation:valueForProgress: (page 3560), thereby saving on the overhead of using a delegate. The current value can be less than 0.0 or greater than 1.0. For example, if you make the value greater than 1.0 you can achieve a "rubber effect" where the size of a view is temporarily larger before its final size.

Availability
Available in Mac OS X v10.4 and later.

See Also
- currentProgress (page 104)
- setAnimationCurve: (page 108)

Declared In
NSAnimation.h

delegate

Returns the delegate of the receiver.

```
- (id < NSAnimationDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setDelegate: (page 109)

Declared In
NSAnimation.h

duration

Returns the duration of the animation, in seconds.

```
- (NSTimeInterval)duration
```

Return Value
An NSTimeInterval value indicating the duration.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setDuration: (page 110)

Declared In
NSAnimation.h

frameRate

Returns the frame rate of the animation.

```
- (float)frameRate
```

Discussion
The frame rate is the number of updates per second. It is not guaranteed to be accurate because of differences between systems on the time needed to process a frame.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSAnimation.h

initWithDuration:animationCurve:

Returns an NSAnimation object initialized with the specified duration and animation-curve values.

```
- (id)initWithDuration:(NSTimeInterval)duration
    animationCurve:(NSAnimationCurve)animationCurve
```

Parameters

duration

The number of seconds over which the animation occurs. Specifying a negative number raises an exception.

animationCurve

An NSAnimationCurve constant that describes the relative speed of the animation over its course; if it is zero, the default curve (NSAnimationEaseInOut) is used.

Return Value
An initialized NSAnimation instance. Returns nil if the object could not be initialized.

Discussion
You can always later change the duration of an NSAnimation object by sending it a setDuration: (page 110) message, even while the animation is running. See "Constants" for descriptions of the NSAnimationCurve constants.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
Reducer

Declared In
NSAnimation.h

isAnimating

Returns a Boolean value that indicates whether the receiver is currently animating.

```
- (BOOL)isAnimating
```

Return Value
YES if the receiver is animating, NO otherwise.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSAnimation.h

progressMarks

Returns the receiver's progress marks.

```
- (NSArray *)progressMarks
```

Return Value
An array of `NSNumber` objects, each encapsulating a `float` value (typed as `NSAnimationProgress`) that represents a current progress mark. If the receiver has no progress marks, an empty array is returned.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `addProgressMark:` (page 102)
- `setProgressMarks:` (page 111)

Declared In
`NSAnimation.h`

removeProgressMark:

Removes progress mark from the receiver.

```
- (void)removeProgressMark:(NSAnimationProgress)progressMark
```

Parameters
progressMark
A `float` value (typed as `NSAnimationProgress`) that indicates the portion of the animation completed. The value should correspond to a progress mark set with addProgressMark: (page 102) or setProgressMarks: (page 111).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `addProgressMark:` (page 102)

Declared In
`NSAnimation.h`

runLoopModesForAnimating

Overridden to return the run-loop modes that the receiver uses to run the animation timer in.

```
- (NSArray *)runLoopModesForAnimating
```

Return Value
An array of constants that indicate the modes the animation's run loop can be in. By default, the method returns `nil`, which indicates that the animation can be run in default, modal, or event-tracking mode. See the `NSRunLoop` class reference for information about the mode constants

Discussion
The value returned from this method is ignored if the animation blocking mode is something other than `NSAnimationNonblocking`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAnimationBlockingMode:` (page 108)

Declared In
`NSAnimation.h`

setAnimationBlockingMode:

Sets the blocking mode of the receiver.

```
- (void)setAnimationBlockingMode:(NSAnimationBlockingMode)animationBlockingMode
```

Parameters

animationBlockingMode

A constant representing the blocking mode the animation is next scheduled to run under. See "NSAnimationBlockingMode" (page 114) for valid values.

If the constant is `NSAnimationNonblocking`, the animation runs in the main thread in one of the standard run-loop modes or in a mode returned from `runLoopModesForAnimating` (page 107). If *animationBlockingMode* is `NSAnimationNonblockingThreaded`, a new thread is spawned to run the animation.

Discussion
The default mode is `NSAnimationBlocking`, which means that the animation runs on the main thread in a custom run-loop mode that blocks user events. The new blocking mode takes effect the next time the receiver is started and has no effect on an animation underway.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- animationBlockingMode` (page 102)

Related Sample Code
From A View to A Movie
Reducer

Declared In
`NSAnimation.h`

setAnimationCurve:

Sets the receiver's animation curve.

```
- (void)setAnimationCurve:(NSAnimationCurve)curve
```

Parameters

curve

An `NSAnimationCurve` constant specifying the animation curve. Invalid values raise an exception.

Discussion

The animation curve describes the relative frame rate over the course of the animation; predefined curves are linear, ease in (slow down near end), ease out (slowly speed up at start), and ease in-ease out (S-curve). Sending this message affects animations already in progress. The `NSAnimationCurve` setting is ignored if the delegate implements `animation:valueForProgress:` (page 3560). See "NSAnimationCurve" (page 113) for valid `NSAnimationCurve` constants.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

From A View to A Movie

QTCoreVideo301

Declared In

`NSAnimation.h`

setCurrentProgress:

Sets the current progress of the receiver.

```
- (void)setCurrentProgress:(NSAnimationProgress)progress
```

Parameters

progress

A `float` value typed as `NSAnimationProgress` that specifies the current progress of the animation. This value should be between 0.0 and 1.0; values that are out of range are pinned to 0.0 or 1.0, whichever is closer.

Discussion

You can use this method to adjust the progress of a running animation. The `NSAnimation` class invokes this method while the animation is running to change the progress for the next frame. Subclasses can override this method to get the latest value and perform their action with it, possibly in a secondary thread. Alternatively, you can implement the delegation method `animation:valueForProgress:` (page 3560).

Availability

Available in Mac OS X v10.4 and later.

See Also

`- currentProgress` (page 104)

Declared In

`NSAnimation.h`

setDelegate:

Sets the delegate of the receiver.

```
- (void)setDelegate:(id < NSAnimationDelegate >)delegate
```

Parameters

delegate

The delegate for the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- delegate (page 105)

Declared In
NSAnimation.h

setDuration:

Sets the duration of the animation to a specified number of seconds.

```
- (void)setDuration:(NSTimeInterval)duration
```

Parameters

duration

An NSTimeInterval value specifying the duration of the animation. Negative values raise an exception.

Discussion
You can change the duration of an animation while it is running. However, setting the duration of a running animation to an interval shorter than the current progress ends the animation.

Availability
Available in Mac OS X v10.4 and later.

See Also
- duration (page 105)

Related Sample Code
From A View to A Movie
QTCoreVideo301
Reducer

Declared In
NSAnimation.h

setFrameRate:

Sets the frame rate of the receiver.

```
- (void)setFrameRate:(float)framesPerSecond
```

Parameters

framesPerSecond

A float value specifying the number of updates per second for the animation. This value must be positive; negative values raise an exception. A frame rate of 0.0 means to go as fast as possible.

Discussion
The frame rate is not guaranteed due to differences among systems for the time needed to process a frame. You can change the frame rate while an animation is running and the new value is used at the next frame. The default frame rate is set to a reasonable value (which is subject to future change).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `frameRate` (page 105)

Declared In
`NSAnimation.h`

setProgressMarks:

Sets the receiver's progress marks to the values specified in the passed-in array.

```
- (void)setProgressMarks:(NSArray *)progressMarks
```

Parameters

progressMarks
An array of `NSNumber` objects, each encapsulating a `float` value (typed as `NSAnimationProgress`) that represents a current progress mark. Passing in `nil` clears all progress marks.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `progressMarks` (page 107)

Declared In
`NSAnimation.h`

startAnimation

Starts the animation represented by the receiver.

```
- (void)startAnimation
```

Discussion
The receiver retains itself and is then autoreleased at the end of the animation or when it receives `stopAnimation` (page 112). If the blocking mode is `NSAnimationBlocking`, the method only returns after the animation has completed or the delegate sends it `stopAnimation` (page 112). If the receiver has a progress of 1.0, it starts again at 0.0.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `startWhenAnimation:reachesProgress:` (page 112)
- `stopAnimation` (page 112)

Related Sample Code
From A View to A Movie
QTCoreVideo301
Reducer

Declared In
NSAnimation.h

startWhenAnimation:reachesProgress:

Starts running the animation represented by the receiver when another animation reaches a specific progress mark.

```
- (void)startWhenAnimation:(NSAnimation *)animation
    reachesProgress:(NSAnimationProgress)startProgress
```

Parameters

animation
The other NSAnimation object with which the receiver is linked.

startProgress
A float value (typed as NSAnimationProgress) that specifies a progress mark of the other animation.

Discussion
This method links the running of two animations together. You can set only one NSAnimation object as a start animation and one as a stop animation at any one time. Setting a new start animation removes any animation previously set.

Availability
Available in Mac OS X v10.4 and later.

See Also
- clearStartAnimation (page 103)
- startAnimation (page 111)
- stopWhenAnimation:reachesProgress: (page 113)

Declared In
NSAnimation.h

stopAnimation

Stops the animation represented by the receiver.

```
- (void)stopAnimation
```

Discussion
The current progress of the receiver is not reset. When this method is sent to instances of NSViewAnimation (a subclass of NSAnimation) the receiver moves to the end frame location.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `startAnimation` (page 111)
- `stopWhenAnimation:reachesProgress:` (page 113)

Related Sample Code
From A View to A Movie

Declared In
`NSAnimation.h`

stopWhenAnimation:reachesProgress:

Stops running the animation represented by the receiver when another animation reaches a specific progress mark.

```
- (void)stopWhenAnimation:(NSAnimation *)animation
    reachesProgress:(NSAnimationProgress)stopProgress
```

Parameters

animation
The other `NSAnimation` object with which the receiver is linked.

stopProgress
A `float` value (typed as `NSAnimationProgress`) that specifies a progress mark of the other animation.

Discussion
This method links the running of two animations together. You can set only one `NSAnimation` object as a start animation and one as a stop animation at any one time. Setting a new stop animation removes any animation previously set.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `clearStopAnimation` (page 103)
- `startWhenAnimation:reachesProgress:` (page 112)
- `stopAnimation` (page 112)

Declared In
`NSAnimation.h`

Constants

NSAnimationCurve

These constants describe the curve of an animation—that is, the relative speed of an animation from start to finish.

```
enum {
    NSAnimationEaseInOut,
    NSAnimationEaseIn,
    NSAnimationEaseOut,
    NSAnimationLinear
};
typedef NSUInteger NSAnimationCurve;
```

Constants

`NSAnimationEaseInOut`

Describes an S-curve in which the animation slowly speeds up and then slows down near the end of the animation. This constant is the default.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

`NSAnimationEaseIn`

Describes an animation that slows down as it reaches the end.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

`NSAnimationEaseOut`

Describes an animation that slowly speeds up from the start.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

`NSAnimationLinear`

Describes an animation in which there is no change in frame rate.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

Discussion

You initialize an `NSAnimation` object using one of these constants with `initWithDuration:animationCurve:` (page 106) and you can set it thereafter with `setAnimationCurve:` (page 108).

NSAnimationBlockingMode

These constants indicate the blocking mode of an `NSAnimation` object when it is running.

```
enum {
    NSAnimationBlocking,
    NSAnimationNonblocking,
    NSAnimationNonblockingThreaded
};
typedef NSUInteger NSAnimationBlockingMode;
```

Constants

`NSAnimationBlocking`

Requests the animation to run in the main thread in a custom run-loop mode that blocks user input.

This is the default.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

`NSAnimationNonblocking`
Requests the animation to run in a standard or specified run-loop mode that allows user input.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

`NSAnimationNonblockingThreaded`
Requests the animation to run in a separate thread that is spawned by the `NSAnimation` object.

The secondary thread has its own run loop.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

Discussion
You specify one of these constants in `setAnimationBlockingMode:` (page 108).

NSAnimationProgress

`NSAnimationProgress` is returned in the userInfo dictionary of an `NSAnimationProgressMarkNotification` (page 116) notification. It will have a value between 0.0 and 1.0

```
typedef float NSAnimationProgress;
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSAnimation.h`

Animation action triggers

These constants are used by the `NSAnimatablePropertyContainer` methods `defaultAnimationForKey:` (page 3554) and `animationForKey:` (page 3555).

```
NSString *NSAnimationTriggerOrderIn;
NSString *NSAnimationTriggerOrderOut;
```

Constants
`NSAnimationTriggerOrderIn`
The trigger that represents the action taken when a view becomes visible, either as a result of being inserted into the visible view hierarchy or the view is no longer set as hidden.

Available in Mac OS X v10.5 and later.

Declared in `NSAnimation.h`.

`NSAnimationTriggerOrderOut`
The trigger that represents the action taken when the view is either removed from the view hierarchy or is hidden.

Available in Mac OS X v10.5 and later.

Declared in `NSAnimation.h`.

NSAnimationProgressMark Notification Key

This constant is returned in the userInfo dictionary of the `NSAnimationProgressMarkNotification` (page 116) notification.

```
NSString*    NSAnimationProgressMark;
```

Constants

`NSAnimationProgressMark`

Contains the value of an `NSAnimationProgress` (page 115) as an `NSNumber` instance that indicates the current animation progress. The value will be between 0.0 and 1.0.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

Notifications

NSAnimationProgressMarkNotification

Posted when the current progress of a running animation reaches one of its progress marks.

The notification object is a running `NSAnimation` object. The `userInfo` dictionary contains the current progress mark, accessed via the key `NSAnimationProgressMark`.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- animation:didReachProgressMark:` (page 3560) (NSAnimationDelegate)

Declared In

`NSAnimation.h`

NSAnimationContext Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSAnimationContext.h
Related sample code	AnimatedTableView BasicCocoaAnimations CocoaSlides QuickLookDownloader TrackBall

Overview

`NSAnimationContext` is analogous to `CATransaction` and are similar in overall concept to `NSGraphicsContext`. Each thread maintains its own stack of nestable `NSAnimationContext` instances, with each new instance initialized as a copy of the instance below (so, inheriting its current properties).

Multiple `NSAnimationContext` instances can be nested, allowing a given block of code to initiate animations using its own specified duration without affecting animations initiated by surrounding code.

```
[NSAnimationContext beginGrouping];
// Animate enclosed operations with a duration of 1 second
[[NSAnimationContext currentContext] setDuration:1.0];
[[aView animator] setFrame:newFrame];
...
    [NSAnimationContext beginGrouping];
    // Animate alpha fades with half-second duration
    [[NSAnimationContext currentContext] setDuration:0.5];
    [[aView animator] setAlphaValue:0.75];
    [[bView animator] setAlphaValue:0.75];
    [NSAnimationContext endGrouping];
...
// Will animate with a duration of 1 second
[[bView animator] setFrame:secondFrame];
[NSAnimationContext endGrouping];
```

Tasks

Grouping Transactions

`+ beginGrouping` (page 118)
Creates a new animation grouping.

`+ endGrouping` (page 119)
Ends the current animation grouping.

Getting the Current Animation Context

`+ currentContext` (page 119)
Returns the current animation context.

Modifying the Animation Duration

`- setDuration:` (page 120)
Sets the duration used by animations created as a result of setting new values for an animatable property.

`- duration` (page 119)
Returns the duration used when animating object properties that support animation.

Class Methods

beginGrouping

Creates a new animation grouping.

```
+ (void)beginGrouping
```

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
BasicCocoaAnimations
CocoaSlides
LayerBackedOpenGLView
MethodReplacement
QuickLookDownloader

Declared In
`NSAnimationContext.h`

currentContext

Returns the current animation context.

```
+ (NSAnimationContext *)currentContext
```

Return Value
The current animation context.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
BasicCocoaAnimations
CocoaSlides
MethodReplacement
QuickLookDownloader
UIElementInspector

Declared In
`NSAnimationContext.h`

endGrouping

Ends the current animation grouping.

```
+ (void)endGrouping
```

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
AnimatedTableView
BasicCocoaAnimations
CocoaSlides
LayerBackedOpenGLView
QuickLookDownloader

Declared In
`NSAnimationContext.h`

Instance Methods

duration

Returns the duration used when animating object properties that support animation.

```
- (NSTimeInterval)duration
```

Return Value
The duration in seconds.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSAnimationContext.h`

setDuration:

Sets the duration used by animations created as a result of setting new values for an animatable property.

```
- (void)setDuration:(NSTimeInterval)duration
```

Parameters

duration
The duration in seconds.

Discussion
Any animations that occur as a result of setting the values of animatable properties in the current context will run for this duration.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
BasicCocoaAnimations
CocoaSlides
MethodReplacement
QuickLookDownloader
UIElementInspector

Declared In
`NSAnimationContext.h`

NSAppleScript Additions Reference

Inherits from	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSAppleScriptExtensions.h
Companion guide	Cocoa Scripting Guide

Overview

The Application Kit adds a method to the Foundation Framework's `NSAppleScript` class to handle rich text source. This method becomes part of the `NSAppleScript` class only for those applications that use the Application Kit.

For more information, see `NSAppleScript` in the *Foundation Framework API Reference*.

Tasks

Obtaining Source

`- richTextSource` (page 121)
Returns the syntax-highlighted source code of the receiver if the receiver has been compiled and its source code is available.

Instance Methods

richTextSource

Returns the syntax-highlighted source code of the receiver if the receiver has been compiled and its source code is available.

```
- (NSAttributedString *)richTextSource
```

Discussion
Returns `nil` otherwise. It is possible for an instance of `NSAppleScript` that has been instantiated with `initWithContentsOfURL:error:` to be a script for which the source code is not available, but is nonetheless executable.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSAppleScriptExtensions.h`

NSApplication Class Reference

Inherits from	NSResponder : NSObject
Conforms to	NSUserInterfaceValidations NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSApplication.h AppKit/NSApplicationScripting.h AppKit/NSColorPanel.h AppKit/NSDataLinkPanel.h AppKit/NSFontPanel.h AppKit/NSHelpManager.h AppKit/NSPageLayout.h AppKit/NSRunningApplication.h AppKit/NSUserInterfaceItemSearching.h
Companion guides	Application Architecture Overview Notification Programming Topics Sheet Programming Topics Services Implementation Guide
Related sample code	CoreRecipes ImageKitDemo NumberInput_IMKit_Sample Quartz Composer WWDC 2005 TextEdit SimpleScriptingPlugin

Class at a Glance

An `NSApplication` object manages an application's main event loop in addition to resources used by all of that application's objects.

Principal Attributes

- Delegate
- Key window

- Display context
- List of windows
- Main window

Commonly Used Methods

`keyWindow` (page 150)
Returns an `NSWindow` object representing the key window.

`mainWindow` (page 151)
Returns the application's main window.

`registerServicesMenuSendTypes:returnTypes:` (page 158)
Specifies which services are valid for this application.

`runModalForWindow:` (page 163)
Runs a modal event loop for the specified `NSWindow` object.

Overview

The `NSApplication` class provides the central framework for your application's execution.

Every application must have exactly one instance of `NSApplication` (or a subclass of `NSApplication`). Your program's `main()` function should create this instance by invoking the `sharedApplication` (page 135) class method. After creating the `NSApplication` object, the `main()` function should load your application's main nib file and then start the event loop by sending the `NSApplication` object a `run` (page 162) message. If you create an Application project in Xcode, this `main()` function is created for you. The `main()` function Xcode creates begins by calling a function named `NSApplicationMain()`, which is functionally similar to the following:

```
void NSApplicationMain(int argc, char *argv[]) {
    [NSApplication sharedApplication];
    [NSBundle loadNibNamed:@"myMain" owner:NSApp];
    [NSApp run];
}
```

The `sharedApplication` (page 135) class method initializes the display environment and connects your program to the window server and the display server. The `NSApplication` object maintains a list of all the `NSWindow` objects the application uses, so it can retrieve any of the application's `NSView` objects. `sharedApplication` (page 135) also initializes the global variable `NSApp`, which you use to retrieve the `NSApplication` instance. `sharedApplication` (page 135) only performs the initialization once; if you invoke it more than once, it simply returns the `NSApplication` object it created previously.

`NSApplication` performs the important task of receiving events from the window server and distributing them to the proper `NSResponder` objects. `NSApp` translates an event into an `NSEvent` object, then forwards the `NSEvent` object to the affected `NSWindow` object. All keyboard and mouse events go directly to the `NSWindow` object associated with the event. The only exception to this rule is if the Command key is pressed when a key-down event occurs; in this case, every `NSWindow` object has an opportunity to respond to the event. When an `NSWindow` object receives an `NSEvent` object from `NSApp`, it distributes it to the objects in its view hierarchy.

NSApplication is also responsible for dispatching certain Apple events received by the application. For example, Mac OS X sends Apple events to your application at various times, such as when the application is launched or reopened. NSApplication installs Apple event handlers to handle these events by sending a message to the appropriate object. You can also use the NSAppleEventManager class to register your own Apple event handlers. The applicationWillFinishLaunching: (page 3578) method is generally the best place to do so. For more information on how events are handled and how you can modify the default behavior, including information on working with Apple events in scriptable applications, see How Cocoa Applications Handle Apple Events in *Cocoa Scripting Guide*.

The NSApplication class sets up autorelease pools (instances of the NSAutoreleasePool class) during initialization and inside the event loop—specifically, within its initialization (or sharedApplication (page 135)) and run (page 162) methods. Similarly, the methods the Application Kit adds to NSBundle employ autorelease pools during the loading of nib files. These autorelease pools aren't accessible outside the scope of the respective NSApplication and NSBundle methods. Typically, an application creates objects either while the event loop is running or by loading objects from nib files, so this lack of access usually isn't a problem. However, if you do need to use Cocoa classes within the main() function itself (other than to load nib files or to instantiate NSApplication), you should create an autorelease pool before using the classes and then release the pool when you're done. For more information, see NSAutoreleasePool in the *Foundation Framework Reference*.

The Delegate and Notifications

You can assign a delegate to NSApp. The delegate responds to certain messages on behalf of NSApp. Some of these messages, such as application:openFile: (page 3566), ask the delegate to perform an action. Another message, applicationShouldTerminate: (page 3576), lets the delegate determine whether the application should be allowed to quit. The NSApplication class sends these messages directly to its delegate.

The NSApp also posts notifications to the application's default notification center. Any object may register to receive one or more of the notifications posted by NSApp by sending the message addObserver:selector:name:object: to the default notification center (an instance of the NSNotificationCenter class). The delegate of NSApp is automatically registered to receive these notifications if it implements certain delegate methods. For example, NSApp posts notifications when it is about to be done launching the application and when it is done launching the application (NSApplicationWillFinishLaunchingNotification (page 195) and NSApplicationDidFinishLaunchingNotification (page 194)). The delegate has an opportunity to respond to these notifications by implementing the methods applicationWillFinishLaunching: (page 3578) and applicationDidFinishLaunching: (page 3571). If the delegate wants to be informed of both events, it implements both methods. If it needs to know only when the application is finished launching, it implements only applicationDidFinishLaunching: (page 3571).

System Services

NSApplication interacts with the system services architecture to provide services to your application through the Services menu.

Subclassing Notes

You rarely should find a real need to create a custom `NSApplication` subclass. Unlike some object-oriented libraries, Cocoa does not require you to create a custom application class to customize application behavior. Instead it gives you many other ways to customize an application. This section discusses both some of the possible reasons to subclass `NSApplication` and some of the reasons *not* to subclass `NSApplication`.

To use a custom subclass of `NSApplication`, simply send `sharedApplication` (page 135) to your subclass rather than directly to `NSApplication`. If you create your application in Xcode, you can accomplish this by setting your custom application class to be the principal class. In Xcode, double-click the application target in the Groups and Files list to open the Info window for the target. Then display the Properties pane of the window and replace "NSApplication" in the Principal Class field with the name of your custom class. The `NSApplicationMain` function sends `sharedApplication` (page 135) to the principal class to obtain the global application instance (`NSApp`)—which in this case will be an instance of your custom subclass of `NSApplication`.

Important: Many Application Kit classes rely on the `NSApplication` class and may not work properly until this class is fully initialized. As a result, you should not, for example, attempt to invoke methods of other Application Kit classes from an initialization method of an `NSApplication` subclass.

Methods to Override

Generally, you subclass `NSApplication` to provide your own special responses to messages that are routinely sent to the global application object (`NSApp`). `NSApplication` does not have primitive methods in the sense of methods that you must override in your subclass. Here are four methods that are possible candidates for overriding:

- Override `run` (page 162) if you want the application to manage the main event loop differently than it does by default. (This a critical and complex task, however, that you should only attempt with good reason.)
- Override `sendEvent:` (page 167) if you want to change how events are dispatched or perform some special event processing.
- Override `requestUserAttention:` (page 161) if you want to modify how your application attracts the attention of the user (for example, offering an alternative to the bouncing application icon in the Dock).
- Override `targetForAction:` (page 174) to substitute another object for the target of an action message.

Special Considerations

The global application object uses autorelease pools in its `run` (page 162) method; if you override this method, you'll need to create your own autorelease pools.

Do not override `sharedApplication` (page 135). The default implementation, which is essential to application behavior, is too complex to duplicate on your own.

Alternatives to Subclassing

`NSApplication` defines over twenty delegate methods that offer opportunities for modifying specific aspects of application behavior. Instead of making a custom subclass of `NSApplication`, your application delegate may be able to implement one or more of these methods to accomplish your design goals. In general, a better design than subclassing `NSApplication` is to put the code that expresses your application's special behavior into one or more custom objects called controllers. Methods defined in your controllers can be invoked from a small dispatcher object without being closely tied to the global application object. For more about application architectures, see Cocoa Design Patterns and The Core Application Architecture on Mac OS X.

Tasks

Getting the Application

`+ sharedApplication` (page 135)
Returns the application instance, creating it if it doesn't exist yet.

Configuring Applications

`- applicationIconImage` (page 138)
Returns the image used for the receiver's icon.

`- setApplicationIconImage:` (page 168)
Sets the receiver's icon to the specified image.

`- delegate` (page 144)
Returns the receiver's delegate.

`- setDelegate:` (page 169)
Makes the given object the receiver's delegate.

Launching Applications

`- finishLaunching` (page 147)
Activates the receiver, opens any files specified by the `NSOpen` user default, and unhighlights the application's icon.

Terminating Applications

`- terminate:` (page 176)
Terminates the receiver.

`- replyToApplicationShouldTerminate:` (page 160)
Responds to `NSTerminateLater` once the application knows whether it can terminate.

Managing Active Status

- `isActive` (page 149)
  Returns a Boolean value indicating whether this is the active application.
- `activateIgnoringOtherApps:` (page 136)
  Makes the receiver the active application.
- `deactivate` (page 143)
  Deactivates the receiver.

Hiding Applications

- `hideOtherApplications:` (page 148)
  Hides all applications, except the receiver.
- `unhideAllApplications:` (page 178)
  Unhides all applications, including the receiver.

Managing the Event Loop

- `isRunning` (page 150)
  Returns a Boolean value indicating whether the main event loop is running.
- `run` (page 162)
  Starts the main event loop.
- `stop:` (page 173)
  Stops the main event loop.
- `runModalForWindow:` (page 163)
  Starts a modal event loop for a given window.
- `stopModal` (page 174)
  Stops a modal event loop.
- `stopModalWithCode:` (page 174)
  Stops a modal event loop, allowing you to return a custom result code.
- `abortModal` (page 135)
  Aborts the event loop started by `runModalForWindow:` (page 163) or `runModalSession:` (page 164).
- `beginModalSessionForWindow:` (page 139)
  Sets up a modal session with the given window and returns an `NSModalSession` structure representing the session.
- `runModalSession:` (page 164)
  Runs a given modal session, as defined in a previous invocation of `beginModalSessionForWindow:`.
- `modalWindow` (page 153)
  Returns the modal window that the receiver is displaying.
- `endModalSession:` (page 146)
  Finishes a modal session.
- `sendEvent:` (page 167)
  Dispatches an event to other objects.

Handling Events

- `currentEvent` (page 142)
  Returns the current event, the last event the receiver retrieved from the event queue.
- `nextEventMatchingMask:untilDate:inMode:dequeue:` (page 153)
  Returns the next event matching a given mask, or `nil` if no such event is found before a specified expiration date.
- `discardEventsMatchingMask:beforeEvent:` (page 144)
  Removes all events matching the given mask and generated before the specified event.

Posting Events

- `postEvent:atStart:` (page 157)
  Adds a given event to the receiver's event queue.

Managing Sheets

- `beginSheet:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 140)
  Starts a document modal session.
- `endSheet:` (page 146)
  Ends a document modal session by specifying the sheet window.
- `endSheet:returnCode:` (page 146)
  Ends a document modal session by specifying the sheet window.

Managing Windows

- `keyWindow` (page 150)
  Returns the window that currently receives keyboard events.
- `mainWindow` (page 151)
  Returns the main window.
- `windowWithWindowNumber:` (page 182)
  Returns the window corresponding to the specified window number.
- `windows` (page 181)
  Returns an array containing the receiver's window objects.
- `makeWindowsPerform:inOrder:` (page 152)
  Sends the specified message to each of the application's window objects until one returns a non-`nil` value.

Minimizing Windows

- `miniaturizeAll:` (page 152)
  Miniaturizes all the receiver's windows.

User Interface Layout Direction

- `userInterfaceLayoutDirection` (page 180)
  Returns the layout direction of the user interface.

Hiding Windows

- `isHidden` (page 150)
  Returns a Boolean value indicating whether the receiver is hidden.
- `hide:` (page 148)
  Hides all the receiver's windows, and the next application in line is activated.
- `unhide:` (page 177)
  Restores hidden windows to the screen and makes the receiver active.
- `unhideWithoutActivation` (page 178)
  Restores hidden windows without activating their owner (the receiver).

Updating Windows

- `updateWindows` (page 179)
  Sends an `update` (page 3405) message to each onscreen window.
- `setWindowsNeedUpdate:` (page 172)
  Sets whether the receiver's windows need updating when the receiver has finished processing the current event.

Managing Window Layers

- `preventWindowOrdering` (page 158)
  Suppresses the usual window ordering in handling the most recent mouse-down event.
- `arrangeInFront:` (page 138)
  Arranges windows listed in the Window menu in front of all other windows.

Accessing the Main Menu

- `mainMenu` (page 151)
  Returns the receiver's main menu.
- `setMainMenu:` (page 170)
  Makes the given menu the receiver's main menu.

Managing the Window Menu

- `windowsMenu` (page 181)
  Returns the Window menu of the application.

- `setWindowsMenu:` (page 172)
  Makes the given menu the receiver's Window menu.
- `addWindowsItem:title:filename:` (page 137)
  Adds an item to the Window menu for a given window.
- `changeWindowsItem:title:filename:` (page 141)
  Changes the item for a given window in the Window menu to a given string.
- `removeWindowsItem:` (page 160)
  Removes the Window menu item for a given window.
- `updateWindowsItem:` (page 179)
  Updates the Window menu item for a given window to reflect the edited status of that window.

Accessing the Dock Tile

- `dockTile` (page 145)
  Returns the application's Dock tile.

Managing the Services Menu

- `registerServicesMenuSendTypes:returnTypes:` (page 158)
  Registers the pasteboard types the receiver can send and receive in response to service requests.
- `servicesMenu` (page 167)
  Returns the Services menu.
- `setServicesMenu:` (page 171)
  Makes a given menu the receiver's Services menu.

Providing Services

- `validRequestorForSendType:returnType:` (page 180)
  Indicates whether the receiver can send and receive the specified pasteboard types.
- `servicesProvider` (page 167)
  Returns the object that provides the services the receiver advertises in the Services menu of other applications.
- `setServicesProvider:` (page 171)
  Registers a given object as the service provider.

Managing Panels

- `orderFrontColorPanel:` (page 155)
  Brings up the color panel, an instance of `NSColorPanel`.
- `orderFrontStandardAboutPanel:` (page 156)
  Displays a standard About window.
- `orderFrontStandardAboutPanelWithOptions:` (page 156)
  Displays a standard About window with information from a given options dictionary.

- `orderFrontCharacterPalette:` (page 155)
  Opens the character palette.
- `runPageLayout:` (page 165)
  Displays the receiver's page layout panel, an instance of `NSPageLayout`.

Displaying Help

- `showHelp:` (page 172)
  If your project is properly registered, and the necessary keys have been set in the property list, this method launches Help Viewer and displays the first page of your application's help book.
- `activateContextHelpMode:` (page 136)
  Places the receiver in context-sensitive help mode.
- `helpMenu` (page 147)
  Returns the help menu used by the application.
- `setHelpMenu:` (page 169)
  Sets the application's help menu.

Managing Threads

+ `detachDrawingThread:toTarget:withObject:` (page 134)
  Creates and executes a new thread based on the specified target and selector.

Posting Actions

- `tryToPerform:with:` (page 177)
  Dispatches an action message to the specified target.
- `sendAction:to:from:` (page 166)
  Sends the given action message to the given target.
- `targetForAction:` (page 174)
  Returns the object that receives the action message specified by the given selector
- `targetForAction:to:from:` (page 175)
  Finds an object that can receive the message specified by the given selector.

Drawing Windows

- `context` (page 142)
  Returns the receiver's display context.

Logging Exceptions

- `reportException:` (page 161)
  Logs a given exception by calling `NSLog()`.

Scripting

- `application:delegateHandlesKey:` (page 182) *delegate method*
  Sent by Cocoa's built-in scripting support during execution of `get` or `set` script commands to find out if the delegate can handle operations on the specified key-value key.
- `orderedDocuments` (page 154)
  Returns an array of document objects arranged according to the front-to-back ordering of their associated windows.
- `orderedWindows` (page 155)
  Returns an array of window objects arranged according to their front-to-back ordering on the screen.

Managing User Attention Requests

- `requestUserAttention:` (page 161)
  Starts a user attention request.
- `cancelUserAttentionRequest:` (page 141)
  Cancels a previous user attention request.
- `replyToOpenOrPrint:` (page 161)
  Handles errors that might occur when the user attempts to open or print files.

Keyboard Accessibility

- `isFullKeyboardAccessEnabled` (page 149)
  Returns that status of Full Keyboard Access set in the Keyboard preference pane.

Presentation Options

- `currentSystemPresentationOptions` (page 143)
  Returns the set of application presentation options that are currently in effect for the system.
- `presentationOptions` (page 158)
  Returns the presentation options that should be in effect for the system when this application is active.
- `setPresentationOptions:` (page 170)
  Sets the application presentation options to use when this application is active.

Activation Policy

- `activationPolicy` (page 137)
  Returns the application's activation policy.
- `setActivationPolicy:` (page 168)
  Attempts to modify the application's activation policy.

Deprecated

- `application:printFiles:` (page 183) *delegate method*
  (Deprecated. Use `application:printFiles:withSettings:showPrintPanels:` (page 3569) instead.)
- `beginModalSessionForWindow:relativeToWindow:` (page 140) Deprecated in Mac OS X v10.0
  (Deprecated. Use `beginSheet:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 140) instead.)
- `runModalForWindow:relativeToWindow:` (page 163) Deprecated in Mac OS X v10.0
  (Deprecated. Use `beginSheet:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 140) instead.)

Spotlight for Help

- `registerUserInterfaceItemSearchHandler:` (page 159)
  Register an an object that provides help data to your application.
- `searchString:inUserInterfaceItemString:searchRange:foundRange:` (page 165)
  Searches for the string in the user interface.
- `unregisterUserInterfaceItemSearchHandler:` (page 178)
  Unregister an an object that provides help data to your application.

Class Methods

detachDrawingThread:toTarget:withObject:

Creates and executes a new thread based on the specified target and selector.

```
+ (void)detachDrawingThread:(SEL)selector toTarget:(id)target withObject:(id)argument
```

Parameters

selector
The selector whose code you want to execute in the new thread.

target
The object that defines the specified selector.

argument
An optional argument you want to pass to the selector.

Discussion
This method is a convenience wrapper for the `detachNewThreadSelector:toTarget:withObject:` method of `NSThread`. This method automatically creates an `NSAutoreleasePool` object for the new thread before invoking *selector*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSApplication.h

sharedApplication

Returns the application instance, creating it if it doesn't exist yet.

```
+ (NSApplication *)sharedApplication
```

Return Value
The shared application object.

Discussion
This method also makes a connection to the window server and completes other initialization. Your program should invoke this method as one of the first statements in `main()`; this invoking is done for you if you create your application with Xcode. To retrieve the `NSApplication` instance after it has been created, use the global variable `NSApp` or invoke this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `run` (page 162)
- `terminate:` (page 176)

Related Sample Code
CoreRecipes
ImageClient
NumberInput_IMKit_Sample
Quartz Composer WWDC 2005 TextEdit
WhackedTV

Declared In
NSApplication.h

Instance Methods

abortModal

Aborts the event loop started by `runModalForWindow:` (page 163) or `runModalSession:` (page 164).

```
- (void)abortModal
```

Discussion
When stopped with this method, `runModalForWindow:` and `runModalSession:` return `NSRunAbortedResponse`.

`abortModal` must be used instead of `stopModal` (page 174) or `stopModalWithCode:` (page 174) when you need to stop a modal event loop from anywhere other than a callout from that event loop. In other words, if you want to stop the loop in response to a user's actions within the modal window, use `stopModal`;

otherwise, use `abortModal`. For example, use `abortModal` when running in a different thread from the Application Kit's main thread or when responding to an `NSTimer` that you have added to the `NSModalPanelRunLoopMode` mode of the default `NSRunLoop`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- endModalSession:` (page 146)

Declared In
`NSApplication.h`

activateContextHelpMode:

Places the receiver in context-sensitive help mode.

```
- (void)activateContextHelpMode:(id)sender
```

Parameters

sender
The object that sent the command.

Discussion
In this mode, the cursor becomes a question mark, and help appears for any user interface item the user clicks.

Most applications don't use this method. Instead, applications enter context-sensitive mode when the user presses the Help key. Applications exit context-sensitive help mode upon the first event after a help window is displayed.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- showHelp:` (page 172)

Declared In
`NSHelpManager.h`

activateIgnoringOtherApps:

Makes the receiver the active application.

```
- (void)activateIgnoringOtherApps:(BOOL)flag
```

Parameters

flag
If `NO`, the application is activated only if no other application is currently active. If `YES`, the application activates regardless.

Discussion
The *flag* parameter is normally set to NO. When the Finder launches an application, using a value of NO for *flag* allows the application to become active if the user waits for it to launch, but the application remains unobtrusive if the user activates another application. Regardless of the setting of *flag*, there may be a time lag before the application activates—you should not assume the application will be active immediately after sending this message.

You rarely need to invoke this method. Under most circumstances, the Application Kit takes care of proper activation. However, you might find this method useful if you implement your own methods for inter-application communication.

You don't need to send this message to make one of the application's NSWindows key. When you send a makeKeyWindow (page 3345) message to an NSWindow object, you ensure that it is the key window when the application is active.

Availability
Available in Mac OS X v10.0 and later.

See Also
- deactivate (page 143)
- isActive (page 149)

Declared In
NSApplication.h

activationPolicy

Returns the application's activation policy.

```
- (NSApplicationActivationPolicy)activationPolicy
```

Return Value
The application's current activation policy.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setActivationPolicy: (page 168)

Declared In
NSApplication.h

addWindowsItem:title:filename:

Adds an item to the Window menu for a given window.

```
- (void)addWindowsItem:(NSWindow *)aWindow title:(NSString *)aString
    filename:(BOOL)isFilename
```

Parameters

aWindow

The window being added to the menu. If this window object already exists in the Window menu, this method has no effect.

aString

The string to display for the window's menu item. How the string is interpreted is dependent on the value in the *isFilename* parameter.

isFilename

If NO, *aString* appears literally in the menu; otherwise, *aString* is assumed to be a converted pathname with the name of the file preceding the path (the way the NSWindow method setTitleWithRepresentedFilename: (page 3399) shows a title)

Discussion

You rarely need to invoke this method directly because Cocoa places an item in the Window menu automatically whenever you set the title of an NSWindow object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- changeWindowsItem:title:filename: (page 141)
- setTitle: (page 3398) (NSWindow)

Related Sample Code

QTAudioExtractionPanel

Declared In

NSApplication.h

applicationIconImage

Returns the image used for the receiver's icon.

```
- (NSImage *)applicationIconImage
```

Return Value

An image containing the application's icon.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setApplicationIconImage: (page 168)

Declared In

NSApplication.h

arrangeInFront:

Arranges windows listed in the Window menu in front of all other windows.

```
- (void)arrangeInFront:(id)sender
```

Parameters

sender

The object that sent the command.

Discussion

Windows associated with the application but not listed in the Window menu are not ordered to the front.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addWindowsItem:title:filename:` (page 137)
- `removeWindowsItem:` (page 160)
- `makeKeyAndOrderFront:` (page 3345) (`NSWindow`)

Declared In

`NSApplication.h`

beginModalSessionForWindow:

Sets up a modal session with the given window and returns an `NSModalSession` structure representing the session.

```
- (NSModalSession)beginModalSessionForWindow:(NSWindow *)aWindow
```

Parameters

aWindow

The window for the session.

Return Value

The `NSModalSession` structure that represents the session.

Discussion

In a modal session, the application receives mouse events only if they occur in *aWindow*. The window is made key, and if not already visible is placed onscreen using the `NSWindow` method `center` (page 3307).

The `beginModalSessionForWindow:` method only sets up the modal session. To actually run the session, use `runModalSession:` (page 164). `beginModalSessionForWindow:` should be balanced by `endModalSession:` (page 146). Make sure these two messages are sent within the same exception-handling scope. That is, if you send `beginModalSessionForWindow:` inside an `NS_DURING` construct, you must send `endModalSession:` before `NS_ENDHANDLER`.

If an exception is raised, `beginModalSessionForWindow:` arranges for proper cleanup. Do not use `NS_DURING` constructs to send an `endModalSession:` message in the event of an exception.

A loop using these methods is similar to a modal event loop run with `runModalForWindow:` (page 163), except the application can continue processing between method invocations.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSApplication.h`

beginModalSessionForWindow:relativeToWindow:

(Deprecated in Mac OS X v10.0. Use beginSheet:modalForWindow:modalDelegate:didEndSelector:contextInfo: (page 140) instead.)

```
- (NSModalSession)beginModalSessionForWindow:(NSWindow *)theWindow
    relativeToWindow:(NSWindow *)docWindow
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.0.

Declared In
NSApplication.h

beginSheet:modalForWindow:modalDelegate:didEndSelector:contextInfo:

Starts a document modal session.

```
- (void)beginSheet:(NSWindow *)sheet modalForWindow:(NSWindow *)docWindow
    modalDelegate:(id)modalDelegate didEndSelector:(SEL)didEndSelector
    contextInfo:(void *)contextInfo
```

Parameters

sheet
The window object representing the sheet you want to display.

docWindow
The window object to which you want to attach the sheet.

modalDelegate
The delegate object that defines your *didEndSelector* method. If nil, the method in *didEndSelector* is not called.

didEndSelector
An optional method to call when the sheet's modal session has ended. This method must be defined on the object in the *modalDelegate* parameter and have the following signature:

```
- (void)sheetDidEnd:(NSWindow *)sheet returnCode:(NSInteger)returnCode
contextInfo:(void *)contextInfo;
```

contextInfo
A pointer to the context info you want passed to the *didEndSelector* method when the sheet's modal session ends.

Discussion
This method runs the modal event loop for the specified sheet synchronously. It displays the sheet, makes it key, starts the run loop, and processes events for it. While the application is in the run loop, it does not respond to any other events (including mouse, keyboard, or window-close events) unless they are associated with the sheet. It also does not perform any tasks (such as firing timers) that are not associated with the modal run loop. In other words, this method consumes only enough CPU time to process events and dispatch them to the action methods associated with the modal window.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `endSheet:` (page 146)
- `endSheet:returnCode:` (page 146)

Related Sample Code
IdentitySample
ImageClient
WhackedTV

Declared In
`NSApplication.h`

cancelUserAttentionRequest:

Cancels a previous user attention request.

```
- (void)cancelUserAttentionRequest:(NSInteger)request
```

Parameters

request
The request identifier returned by the `requestUserAttention:` method.

Discussion
A request is also canceled automatically by user activation of the application.

Availability
Available in Mac OS X v10.1 and later.

See Also
- `requestUserAttention:` (page 161)

Declared In
`NSApplication.h`

changeWindowsItem:title:filename:

Changes the item for a given window in the Window menu to a given string.

```
- (void)changeWindowsItem:(NSWindow *)aWindow title:(NSString *)aString
    filename:(BOOL)isFilename
```

Parameters

aWindow
The window whose title you want to change in the Window menu. If *aWindow* is not in the Window menu, this method adds it.

aString
The string to display for the window's menu item. How the string is interpreted is dependent on the value in the *isFilename* parameter.

isFilename
If NO, *aString* appears literally in the menu; otherwise, *aString* is assumed to be a converted pathname with the name of the file preceding the path (the way the NSWindow method setTitleWithRepresentedFilename: (page 3399) shows a title)

Availability
Available in Mac OS X v10.0 and later.

See Also
- addWindowsItem:title:filename: (page 137)
- removeWindowsItem: (page 160)
- setTitle: (page 3398) (NSWindow)

Declared In
NSApplication.h

context

Returns the receiver's display context.

```
- (NSGraphicsContext *)context
```

Return Value
The current display context for the application.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSApplication.h

currentEvent

Returns the current event, the last event the receiver retrieved from the event queue.

```
- (NSEvent *)currentEvent
```

Return Value
The last event object retrieved by the application.

Discussion
NSApp receives events and forwards them to the affected NSWindow objects, which then distribute them to the objects in its view hierarchy.

Availability
Available in Mac OS X v10.0 and later.

See Also
- discardEventsMatchingMask:beforeEvent: (page 144)
- postEvent:atStart: (page 157)
- sendEvent: (page 167)

Related Sample Code
ClockControl

Declared In
`NSApplication.h`

currentSystemPresentationOptions

Returns the set of application presentation options that are currently in effect for the system.

```
- (NSApplicationPresentationOptions)currentSystemPresentationOptions
```

Return Value
The presentation options. The constants are listed in "NSApplicationPresentationOptions" (page 187) and can combined using a C bitwise OR operator.

Discussion
These are the presentation options that have been put into effect by the currently active application.

This method is key-value observable.

A client that observes `currentSystemPresentationOptions` will receive notifications when

- The client is the active application, and makes a change itself using either `setPresentationOptions:` (page 170) or `SetSystemUIMode`.
- Another application is active, and makes presentation changes of its own.
- Another application becomes active and causes the active set of presentation options to change.

Key-value observing notifications are not sent when one of the above conditions occur, but has the same set of presentation options as the previously active application.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `presentationOptions` (page 158)
- `setPresentationOptions` (page 170)

Declared In
`NSApplication.h`

deactivate

Deactivates the receiver.

```
- (void)deactivate
```

Discussion
Normally, you shouldn't invoke this method—the Application Kit is responsible for proper deactivation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `activateIgnoringOtherApps:` (page 136)

Declared In
`NSApplication.h`

delegate

Returns the receiver's delegate.

```
- (id < NSApplicationDelegate >)delegate
```

Return Value
The application delegate object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDelegate:` (page 169)

Declared In
`NSApplication.h`

discardEventsMatchingMask:beforeEvent:

Removes all events matching the given mask and generated before the specified event.

```
- (void)discardEventsMatchingMask:(NSUInteger)mask beforeEvent:(NSEvent *)lastEvent
```

Parameters

mask
Contains one or more flags indicating the types of events to discard. The constants section of the `NSEvent` class defines the constants you can add together to create this mask. The discussion section also lists some of the constants that are typically used.

lastEvent
A marker event that you use to indicate which events should be discarded. Events that occurred before this event are discarded but those that occurred after it are not.

Discussion
Use this method to ignore any events that occurred before a specific event. For example, suppose your application has a tracking loop that you exit when the user releases the mouse button. You could use this method, specifying `NSAnyEventMask` as the mask argument and the ending mouse-up event as the *lastEvent* argument, to discard all events that occurred while you were tracking mouse movements in your loop. Passing the mouse-up event as *lastEvent* ensures that any events that might have occurred after the mouse-up event (that is, that appear in the queue after the mouse-up event) are not discarded.

Note: Typically, you send this message to an `NSWindow` object, rather than to the application object. Discarding events for a window clears out all of the events for that window only, leaving events for other windows in place.

For the *mask* parameter, you can add together event type constants such as the following:

```
NSLeftMouseDownMask
NSLeftMouseUpMask
NSRightMouseDownMask
NSRightMouseUpMask
NSMouseMovedMask
NSLeftMouseDraggedMask
NSRightMouseDraggedMask
NSMouseEnteredMask
NSMouseExitedMask
NSKeyDownMask
NSKeyUpMask
NSFlagsChangedMask
NSPeriodicMask
NSCursorUpdateMask
NSAnyEventMask
```

This method can also be called in subthreads. Events posted in subthreads bubble up in the main thread event queue.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- nextEventMatchingMask:untilDate:inMode:dequeue:` (page 153)

Declared In
`NSApplication.h`

dockTile

Returns the application's Dock tile.

```
- (NSDockTile *)dockTile
```

Return Value
The application's Dock tile.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSApplication.h`

endModalSession:

Finishes a modal session.

```
- (void)endModalSession:(NSModalSession)session
```

Parameters

session

A modal session structure returned by a previous invocation of `beginModalSessionForWindow:`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `beginModalSessionForWindow:` (page 139)
- `runModalSession:` (page 164)

Declared In
`NSApplication.h`

endSheet:

Ends a document modal session by specifying the sheet window.

```
- (void)endSheet:(NSWindow *)sheet
```

Parameters

sheet

The sheet whose modal session you want to end.

Discussion
This method ends the modal session with the return code `NSRunStoppedResponse`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `beginSheet:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 140)
- `endSheet:returnCode:` (page 146)

Related Sample Code
WhackedTV

Declared In
`NSApplication.h`

endSheet:returnCode:

Ends a document modal session by specifying the sheet window.

```
- (void)endSheet:(NSWindow *)sheet returnCode:(NSInteger)returnCode
```

Parameters

sheet

The sheet whose modal session you want to end.

returnCode

The return code to send to the delegate. You can use one of the return codes defined in "Return values for modal operations" (page 184) or a custom value that you define.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `beginSheet:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 140)
- `endSheet:` (page 146)

Related Sample Code
IdentitySample
ImageClient

Declared In
`NSApplication.h`

finishLaunching

Activates the receiver, opens any files specified by the `NSOpen` user default, and unhighlights the application's icon.

```
- (void)finishLaunching
```

Discussion
The `run` (page 162) method invokes this method before it starts the event loop. When this method begins, it posts an `NSApplicationWillFinishLaunchingNotification` (page 195) to the default notification center. If you override `finishLaunching` (page 147), the subclass method should invoke the superclass method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `applicationWillFinishLaunching:` (page 3578) (NSApplicationDelegate)
- `applicationDidFinishLaunching:` (page 3571) (NSApplicationDelegate)

Related Sample Code
GLUT

Declared In
`NSApplication.h`

helpMenu

Returns the help menu used by the application.

```
- (NSMenu *)helpMenu
```

Return Value
Returns the application's help menu.

Discussion
If `helpMenu` is `nil`, the system will append the help to an appropriate menu and will not return a reference to that menu when this method is called.,

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setHelpMenu:` (page 169)

Declared In
`NSApplication.h`

hide:

Hides all the receiver's windows, and the next application in line is activated.

```
- (void)hide:(id)sender
```

Parameters

sender

The object that sent the command.

Discussion
This method is usually invoked when the user chooses Hide in the application's main menu. When this method begins, it posts an `NSApplicationWillHideNotification` (page 195) to the default notification center. When it completes successfully, it posts an `NSApplicationDidHideNotification` (page 194).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- miniaturizeAll:` (page 152)
`- unhide:` (page 177)
`- unhideWithoutActivation` (page 178)
`- applicationDidHide:` (page 3572) (NSApplicationDelegate)
`- applicationWillHide:` (page 3578) (NSApplicationDelegate)

Declared In
`NSApplication.h`

hideOtherApplications:

Hides all applications, except the receiver.

```
- (void)hideOtherApplications:(id)sender
```

Parameters

sender

The object that sent this message.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSApplication.h

isActive

Returns a Boolean value indicating whether this is the active application.

```
- (BOOL)isActive
```

Return Value
YES if this is the active application; NO otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- activateIgnoringOtherApps: (page 136)
- deactivate (page 143)

Declared In
NSApplication.h

isFullKeyboardAccessEnabled

Returns that status of Full Keyboard Access set in the Keyboard preference pane.

```
- (BOOL)isFullKeyboardAccessEnabled
```

Return Value
YES if Full Keyboard Access is enabled, otherwise NO.

Discussion
You may use this status to implement your own key loop or to implement in-control tabbing behavior similar to NSTableView. Because of the nature of the preference storage, you will not be notified of changes to the key if you attempt to observe it via key-value observing; however, calling this method is fairly inexpensive, so you should always call it when you need the underlying value instead of caching it.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSApplication.h

isHidden

Returns a Boolean value indicating whether the receiver is hidden.

```
- (BOOL)isHidden
```

Return Value
`YES` if the receiver is hidden, `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hide:` (page 148)
- `unhide:` (page 177)
- `unhideWithoutActivation` (page 178)

Declared In
`NSApplication.h`

isRunning

Returns a Boolean value indicating whether the main event loop is running.

```
- (BOOL)isRunning
```

Return Value
`YES` if the main event loop is running; `NO` otherwise.

Discussion
`NO` means the `stop:` (page 173) method was invoked.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `run` (page 162)
- `terminate:` (page 176)

Related Sample Code
GLUT

Declared In
`NSApplication.h`

keyWindow

Returns the window that currently receives keyboard events.

```
- (NSWindow *)keyWindow
```

Return Value
The window object currently receiving keyboard events or `nil` if there is no key window.

Discussion
This method might return `nil` if the application's nib file hasn't finished loading yet or if the receiver is not active.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `mainWindow` (page 151)
- `isKeyWindow` (page 3338) (NSWindow)

Declared In
`NSApplication.h`

mainMenu

Returns the receiver's main menu.

```
- (NSMenu *)mainMenu
```

Return Value
The menu object representing the application's menu bar.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMainMenu:` (page 170)

Declared In
`NSApplication.h`

mainWindow

Returns the main window.

```
- (NSWindow *)mainWindow
```

Return Value
The application's main window or `nil` if there is no main window.

Discussion
This method might return `nil` if the application's nib file hasn't finished loading, if the receiver is not active, or if the application is hidden.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `keyWindow` (page 150)
- `isMainWindow` (page 3338) (NSWindow)

Related Sample Code
PDFKitLinker2

Declared In
NSApplication.h

makeWindowsPerform:inOrder:

Sends the specified message to each of the application's window objects until one returns a non-nil value.

```
- (NSWindow *)makeWindowsPerform:(SEL)aSelector inOrder:(BOOL)flag
```

Parameters

aSelector

The selector to perform on each window. This method must not take any arguments and must return a value whose type that can be compared to nil.

flag

If YES, the *aSelector* message is sent to each of the window server's onscreen windows, going in z-order, until one returns a non-nil value. A minimized window is not considered to be onscreen for this check. If NO, the message is sent to all windows in NSApp's window list, regardless of whether or not they are onscreen. This order is unspecified.

Return Value
The window that returned a non-nil value or nil if all windows returned nil from *aSelector*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- sendAction:to:from: (page 166)
- tryToPerform:with: (page 177)
- windows (page 181)

Declared In
NSApplication.h

miniaturizeAll:

Miniaturizes all the receiver's windows.

```
- (void)miniaturizeAll:(id)sender
```

Parameters

sender

The object that sent the command.

Availability
Available in Mac OS X v10.0 and later.

See Also
- hide: (page 148)

Declared In
NSApplication.h

modalWindow

Returns the modal window that the receiver is displaying.

```
- (NSWindow *)modalWindow
```

Return Value
The modal window being displayed or nil if no modal window is being displayed.

Discussion
This method returns the current standalone modal window. It does not return sheets that are attached to other windows. If you need to retrieve a sheet window, use the attachedSheet (page 3300) method of NSWindow.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSApplication.h

nextEventMatchingMask:untilDate:inMode:dequeue:

Returns the next event matching a given mask, or nil if no such event is found before a specified expiration date.

```
- (NSEvent *)nextEventMatchingMask:(NSUInteger)mask untilDate:(NSDate *)expiration
    inMode:(NSString *)mode dequeue:(BOOL)flag
```

Parameters

mask
Contains one or more flags indicating the types of events to return. The constants section of the NSEvent class defines the constants you can add together to create this mask. The discardEventsMatchingMask:beforeEvent: (page 144) method also lists several of these constants.

expiration
The expiration date for the current event request. Specifying nil for this parameter is equivalent to returning a date object using the distantPast method.

mode
The run loop mode in which to run while looking for events. The mode you specify also determines which timers and run-loop observers may fire while the application waits for the event.

flag
Specify YES if you want the event removed from the queue.

Return Value
The event object whose type matches one of the event types specified by the *mask* parameter.

Discussion
You can use this method to short circuit normal event dispatching and get your own events. For example, you may want to do this in response to a mouse-down event in order to track the mouse while its button is down. (In such an example, you would pass the appropriate event types for mouse-dragged and mouse-up events to the *mask* parameter and specify the `NSEventTrackingRunLoopMode` run loop mode.) Events that do not match one of the specified event types are left in the queue.

You can specify one of the run loop modes defined by the Application Kit or a custom run loop mode used specifically by your application. Application Kit defines the following run-loop modes:

```
NSDefaultRunLoopMode
NSEventTrackingRunLoopMode
NSModalPanelRunLoopMode
NSConnectionReplyMode
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `postEvent:atStart:` (page 157)
- `run` (page 162)
- `runModalForWindow:` (page 163)

Declared In
`NSApplication.h`

orderedDocuments

Returns an array of document objects arranged according to the front-to-back ordering of their associated windows.

```
- (NSArray *)orderedDocuments
```

Return Value
An array of `NSDocument` objects, where the position of a document is based on the front-to-back ordering of its associated window.

Discussion
This method is called during script command evaluation—for example, while finding the document in the script statement `the third rectangle in the first document`. For information on how your application can return its own array of ordered documents, see `application:delegateHandlesKey:`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `orderedWindows` (page 155)

Declared In
`NSApplicationScripting.h`

orderedWindows

Returns an array of window objects arranged according to their front-to-back ordering on the screen.

```
- (NSArray *)orderedWindows
```

Return Value
An array of `NSWindow` objects, where the position of each window in the array corresponds to the front-to-back ordering of the windows on the screen.

Discussion
Only windows that are typically scriptable are included in the returned array. For example, panels are not included.

This method is called during script command evaluation—for example, while finding the window in the script statement `close the second window`. For information on how your application can return its own array of ordered windows, see `application:delegateHandlesKey:`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- orderedDocuments` (page 154)

Declared In
`NSApplicationScripting.h`

orderFrontCharacterPalette:

Opens the character palette.

```
- (void)orderFrontCharacterPalette:(id)sender
```

Parameters
sender
The object that sent the command.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSApplication.h`

orderFrontColorPanel:

Brings up the color panel, an instance of `NSColorPanel`.

```
- (void)orderFrontColorPanel:(id)sender
```

Parameters
sender
The object that sent the command.

Discussion
If the `NSColorPanel` object does not exist yet, this method creates one. This method is typically invoked when the user chooses Colors from a menu.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorPanel.h`

orderFrontStandardAboutPanel:

Displays a standard About window.

```
- (void)orderFrontStandardAboutPanel:(id)sender
```

Parameters

sender
The object that sent the command.

Discussion
This method calls `orderFrontStandardAboutPanelWithOptions:` (page 156) with a `nil` argument. See `orderFrontStandardAboutPanelWithOptions:` for a description of what's displayed.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MenuItemView

Declared In
`NSApplication.h`

orderFrontStandardAboutPanelWithOptions:

Displays a standard About window with information from a given options dictionary.

```
- (void)orderFrontStandardAboutPanelWithOptions:(NSDictionary *)optionsDictionary
```

Parameters

optionsDictionary
A dictionary whose keys define the contents of the About window. See the discussion for a description of the available keys.

Discussion
The following strings are keys that can occur in *optionsDictionary*:

- `@"Credits"`: An NSAttributedString displayed in the info area of the panel. If not specified, this method then looks for a file named "`Credits.html`", "`Credits.rtf`", and "`Credits.rtfd`", in that order, in the bundle returned by the `NSBundle` class method `mainBundle`. The first file found is used. If none is found, the info area is left blank.

- `@"ApplicationName"`: An `NSString` object displayed as the application's name. If not specified, this method then uses the value of `CFBundleName` (localizable). If neither is found, this method uses `[[NSProcessInfo processInfo] processName]`.
- `@"ApplicationIcon"`: An `NSImage` object displayed as the application's icon. If not specified, this method then looks for an image named "`NSApplicationIcon`", using `[NSImage imageNamed:@"NSApplicationIcon"]`. If neither is available, this method uses the generic application icon.
- `@"Version"`: An `NSString` object with the build version number of the application ("58.4"), displayed as "(v58.4)". If not specified, obtain from the `CFBundleVersion` key in `infoDictionary`; if not specified, leave blank (the "(v)" is not displayed).
- `@"Copyright"`: An `NSString` object with a line of copyright information. If not specified, this method then looks for the value of `NSHumanReadableCopyright` in the localized version `infoDictionary`. If neither is available, this method leaves the space blank.
- `@"ApplicationVersion"`: An `NSString` object with the application version ("Mac OS X", "3", "WebObjects 4.5", "AppleWorks 6",...). If not specified, obtain from the `CFBundleShortVersionString` key in `infoDictionary`. If neither is available, the build version, if available, is printed alone, as "Version x.x".

Availability
Available in Mac OS X v10.0 and later.

See Also
`- orderFrontStandardAboutPanel:` (page 156)

Declared In
`NSApplication.h`

postEvent:atStart:

Adds a given event to the receiver's event queue.

```
- (void)postEvent:(NSEvent *)anEvent atStart:(BOOL)flag
```

Parameters

anEvent
The event object to post to the queue.

flag
Specify `YES` to add the event to the front of the queue; otherwise, specify `NO` to add the event to the back of the queue.

Discussion
This method can also be called in subthreads. Events posted in subthreads bubble up in the main thread event queue.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- currentEvent` (page 142)
`- sendEvent:` (page 167)

Declared In
NSApplication.h

presentationOptions

Returns the presentation options that should be in effect for the system when this application is active.

```
- (NSApplicationPresentationOptions)presentationOptions
```

Return Value
The application's presentation options. The constants are listed in "NSApplicationPresentationOptions" (page 187) and they combined using a C bitwise OR operator.

Discussion
This method is key-value observable.

Availability
Available in Mac OS X v10.6 and later.

See Also
- currentSystemPresentationOptions (page 143)
- setPresentationOptions (page 170)

Declared In
NSApplication.h

preventWindowOrdering

Suppresses the usual window ordering in handling the most recent mouse-down event.

```
- (void)preventWindowOrdering
```

Discussion
This method is only useful for mouse-down events when you want to prevent the window that receives the event from being ordered to the front.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSApplication.h

registerServicesMenuSendTypes:returnTypes:

Registers the pasteboard types the receiver can send and receive in response to service requests.

```
- (void)registerServicesMenuSendTypes:(NSArray *)sendTypes returnTypes:(NSArray
    *)returnTypes
```

Parameters

sendTypes

An array of `NSString` objects, each of which corresponds to a particular pasteboard type that the application can send.

returnTypes

An array of `NSString` objects, each of which corresponds to a particular pasteboard type that the application can receive.

Discussion

If the receiver has a Services menu, a menu item is added for each service provider that can accept one of the specified *sendTypes* or return one of the specified *returnTypes*. You should typically invoke this method at application startup time or when an object that can use services is created. You can invoke it more than once—its purpose is to ensure there is a menu item for every service the application can use. The event-handling mechanism will dynamically enable the individual items to indicate which services are currently appropriate. All the `NSResponder` objects in your application (typically `NSView` objects) should register every possible type they can send and receive by sending this message to `NSApp`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `validRequestorForSendType:returnType:` (page 180)
- `readSelectionFromPasteboard:` (page 3797) (NSServicesRequests protocol)
- `writeSelectionToPasteboard:types:` (page 3798) (NSServicesRequests protocol)

Declared In

`NSApplication.h`

registerUserInterfaceItemSearchHandler:

Register an an object that provides help data to your application.

```
- (void)registerUserInterfaceItemSearchHandler:(id < NSUserInterfaceItemSearching
    >)handler
```

Parameters

handler

The class instance that conforms to `NSUserInterfaceItemSearching` and provides help content.

Discussion

You can register as many search handlers as you like. If you register the same instance more than once the subsequent registrations are ignored.

Availability

Available in Mac OS X v10.6 and later.

See Also

- `unregisterUserInterfaceItemSearchHandler:` (page 178)
- `searchString:inUserInterfaceItemString:searchRange:foundRange:` (page 165)

Declared In

`NSUserInterfaceItemSearching.h`

removeWindowsItem:

Removes the Window menu item for a given window.

```
- (void)removeWindowsItem:(NSWindow *)aWindow
```

Parameters

aWindow

The window whose menu item is to be removed.

Discussion

This method doesn't prevent the item from being automatically added again. Use the `setExcludedFromWindowsMenu:` (page 3380) method of `NSWindow` if you want the item to remain excluded from the Window menu.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- addWindowsItem:title:filename:` (page 137)

`- changeWindowsItem:title:filename:` (page 141)

Declared In

`NSApplication.h`

replyToApplicationShouldTerminate:

Responds to `NSTerminateLater` once the application knows whether it can terminate.

```
- (void)replyToApplicationShouldTerminate:(BOOL)shouldTerminate
```

Parameters

shouldTerminate

Specify `YES` if you want the application to terminate; otherwise, specify `NO`.

Discussion

If your application delegate returns `NSTerminateLater` from its `applicationShouldTerminate:` (page 3576) method, your code must subsequently call this method to let the `NSApplication` object know whether it can actually terminate itself.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ExtractMovieAudioToAIFF

QTExtractAndConvertToAIFF

QTExtractAndConvertToMovieFile

Declared In

`NSApplication.h`

replyToOpenOrPrint:

Handles errors that might occur when the user attempts to open or print files.

```
- (void)replyToOpenOrPrint:(NSApplicationDelegateReply)reply
```

Parameters

reply

The error that occurred. For a list of possible values, see "Constants" (page 184).

Discussion

Delegates should invoke this method if an error is encountered in the `application:openFiles:` (page 3566) or `application:printFiles:withSettings:showPrintPanels:` (page 3569) delegate methods.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`NSApplication.h`

reportException:

Logs a given exception by calling `NSLog()`.

```
- (void)reportException:(NSException *)anException
```

Parameters

anException

The exception whose contents you want to write to the log file.

Discussion

This method does not raise *anException*. Use it inside of an exception handler to record that the exception occurred.

Availability

Available in Mac OS X v10.0 and later.

See Also

`NSSetUncaughtExceptionHandler` (Foundation Functions)

Declared In

`NSApplication.h`

requestUserAttention:

Starts a user attention request.

```
- (NSInteger)requestUserAttention:(NSRequestUserAttentionType)requestType
```

Parameters

requestType

The severity of the request. For a list of possible values, see "Constants" (page 184).

Return Value
The identifier for the request. You can use this value to cancel the request later using the `cancelUserAttentionRequest:` method.

Discussion
Activating the application cancels the user attention request. A spoken notification will occur if spoken notifications are enabled. Sending `requestUserAttention:` to an application that is already active has no effect.

If the inactive application presents a modal panel, this method will be invoked with `NSCriticalRequest` automatically. The modal panel is not brought to the front for an inactive application.

Availability
Available in Mac OS X v10.1 and later.

See Also
- `cancelUserAttentionRequest:` (page 141)

Declared In
`NSApplication.h`

run

Starts the main event loop.

```
- (void)run
```

Discussion
The loop continues until a `stop:` (page 173) or `terminate:` (page 176) message is received. Upon each iteration through the loop, the next available event from the window server is stored and then dispatched by sending it to `NSApp` using `sendEvent:` (page 167).

After creating the `NSApplication` object, the `main` function should load your application's main nib file and then start the event loop by sending the `NSApplication` object a `run` message. If you create an Cocoa application project in Xcode, this `main` function is implemented for you.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `runModalForWindow:` (page 163)
- `runModalSession:` (page 164)
- `applicationDidFinishLaunching:` (page 3571) (NSApplicationDelegate)

Related Sample Code
GLUT
NumberInput_IMKit_Sample

Declared In
`NSApplication.h`

runModalForWindow:

Starts a modal event loop for a given window.

```
- (NSInteger)runModalForWindow:(NSWindow *)aWindow
```

Parameters

aWindow

The window to be displayed modally. If it is not already visible, the window is centered on the screen using the value in its `center` (page 3307)method and made visible and key. If it is already visible, it is simply made key.

Return Value

An integer indicating the reason that this method returned. See the discussion for a description of possible return values.

Discussion

This method runs a modal event loop for the specified window synchronously. It displays the specified window, makes it key, starts the run loop, and processes events for that window. (You do not need to show the window yourself.) While the application is in that loop, it does not respond to any other events (including mouse, keyboard, or window-close events) unless they are associated with the window. It also does not perform any tasks (such as firing timers) that are not associated with the modal run loop. In other words, this method consumes only enough CPU time to process events and dispatch them to the action methods associated with the modal window.

You can exit the modal loop by calling the `stopModal`, `stopModalWithCode:`, or `abortModal` methods from your modal window code. If you use the `stopModalWithCode:` method to stop the modal event loop, this method returns the argument passed to `stopModalWithCode:`. If you use `stopModal` instead, this method returns the constant `NSRunStoppedResponse`. If you use `abortModal`, this method returns the constant `NSRunAbortedResponse`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `run` (page 162)
- `runModalSession:` (page 164)

Related Sample Code

WhackedTV

Declared In

`NSApplication.h`

runModalForWindow:relativeToWindow:

(Deprecated in Mac OS X v10.0. Use `beginSheet:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 140) instead.)

```
- (NSInteger)runModalForWindow:(NSWindow *)theWindow relativeToWindow:(NSWindow
    *)docWindow
```

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.0.

Declared In
NSApplication.h

runModalSession:

Runs a given modal session, as defined in a previous invocation of `beginModalSessionForWindow:`.

```
- (NSInteger)runModalSession:(NSModalSession)session
```

Parameters

session

The modal session structure returned by the `beginModalSessionForWindow:` method for the window to be displayed.

Return Value
An integer indicating the reason that this method returned. See the discussion for a description of possible return values.

Discussion
A loop that uses this method is similar in some ways to a modal event loop run with `runModalForWindow:`, except with this method your code can do some additional work between method invocations. When you invoke this method, events for the `NSWindow` object of this session are dispatched as normal. This method returns when there are no more events. You must invoke this method frequently enough in your loop that the window remains responsive to events. However, you should not invoke this method in a tight loop because it returns immediately if there are no events, and consequently you could end up polling for events rather than blocking.

Typically, you use this method in situations where you want to do some additional processing on the current thread while the modal loop runs. For example, while processing a large data set, you might want to use a modal dialog to display progress and give the user a chance to cancel the operation. If you want to display a modal dialog and do not need to do any additional work in parallel, use `runModalForWindow:` instead. When there are no pending events, that method waits idly instead of consuming CPU time.

The following code shows a sample loop you can use in your code:

```
NSModalSession session = [NSApp beginModalSessionForWindow:theWindow];
for (;;) {
    if ([NSApp runModalSession:session] != NSRunContinuesResponse)
        break;
    [self doSomeWork];
}
[NSApp endModalSession:session];
```

If the modal session was not stopped, this method returns `NSRunContinuesResponse`. At this point, your application can do some work before the next invocation of `runModalSession:` (as indicated in the example's `doSomeWork` call). If `stopModal` (page 174) was invoked as the result of event processing, `runModalSession:` returns `NSRunStoppedResponse`. If `stopModalWithCode:` (page 174) was invoked, this method returns the value passed to `stopModalWithCode:`. If `abortModal` (page 135) was invoked, this method returns `NSRunAbortedResponse`.

The window is placed on the screen and made key as a result of the `runModalSession:` message. Do not send a separate `makeKeyAndOrderFront:` (page 3345) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `beginModalSessionForWindow:` (page 139)
- `endModalSession:` (page 146)
- `run` (page 162)
- `runModalForWindow:` (page 163)

Declared In
`NSApplication.h`

runPageLayout:

Displays the receiver's page layout panel, an instance of `NSPageLayout`.

```
- (void)runPageLayout:(id)sender
```

Parameters

sender
The object that sent the command.

Discussion
If the `NSPageLayout` instance does not exist, this method creates one. This method is typically invoked when the user chooses Page Setup from the application's FIle menu.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPageLayout.h`

searchString:inUserInterfaceItemString:searchRange:foundRange:

Searches for the string in the user interface.

```
- (BOOL)searchString:(NSString *)searchString inUserInterfaceItemString:(NSString
    *)stringToSearch searchRange:(NSRange)searchRange foundRange:(NSRange
   *)foundRange
```

Parameters

searchString
The search string.

stringToSearch
The string to search.

searchRange
The subrange of the *stringToSearch* to restrict the search to.

foundRange
Returns, by-reference, the range of the *searchString* within *stringToSearch*.

Return Value
`YES` if the searchString is matched, otherwise `NO`.

Discussion
The search uses the default matching rules for Spotlight for Help.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `registerUserInterfaceItemSearchHandler:` (page 159)

Declared In
`NSUserInterfaceItemSearching.h`

sendAction:to:from:

Sends the given action message to the given target.

```
- (BOOL)sendAction:(SEL)anAction to:(id)aTarget from:(id)sender
```

Parameters

anAction
The action message you want to send.

aTarget
The target object that defines the specified action message.

sender
The object to pass for the action message's parameter.

Return Value
`YES` if the action was successfully sent; otherwise `NO`. This method also returns `NO` if *anAction* is `nil`.

Discussion
If *aTarget* is `nil`, `sharedApplication` (page 135) looks for an object that can respond to the message—that is, an object that implements a method matching *anAction*. It begins with the first responder of the key window. If the first responder can't respond, it tries the first responder's next responder and continues following next responder links up the responder chain. If none of the objects in the key window's responder chain can handle the message, `sharedApplication` (page 135) attempts to send the message to the key window's delegate.

If the delegate doesn't respond and the main window is different from the key window, `sharedApplication` (page 135) begins again with the first responder in the main window. If objects in the main window can't respond, `sharedApplication` (page 135) attempts to send the message to the main window's delegate. If still no object has responded, `sharedApplication` (page 135) tries to handle the message itself. If `sharedApplication` (page 135) can't respond, it attempts to send the message to its own delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `targetForAction:` (page 174)
- `tryToPerform:with:` (page 177)
- `makeWindowsPerform:inOrder:` (page 152)

Declared In
NSApplication.h

sendEvent:

Dispatches an event to other objects.

```
- (void)sendEvent:(NSEvent *)anEvent
```

Parameters

anEvent
The event object to dispatch.

Discussion
You rarely invoke `sendEvent:` directly, although you might want to override this method to perform some action on every event. `sendEvent:` messages are sent from the main event loop (the `run` (page 162) method). `sendEvent:` is the method that dispatches events to the appropriate responders—`NSApp` handles application events, the `NSWindow` object indicated in the event record handles window-related events, and mouse and key events are forwarded to the appropriate `NSWindow` object for further dispatching.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `currentEvent` (page 142)
- `postEvent:atStart:` (page 157)

Declared In
NSApplication.h

servicesMenu

Returns the Services menu.

```
- (NSMenu *)servicesMenu
```

Return Value
The Services menu or `nil` if no Services menu has been created

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setServicesMenu:` (page 171)

Declared In
NSApplication.h

servicesProvider

Returns the object that provides the services the receiver advertises in the Services menu of other applications.

```
- (id)servicesProvider
```

Return Value
The application's service provider object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setServicesProvider: (page 171)

Declared In
NSApplication.h

setActivationPolicy:

Attempts to modify the application's activation policy.

```
- (BOOL)setActivationPolicy:(NSApplicationActivationPolicy)activationPolicy
```

Parameters
activationPolicy
The desired activation policy.

Return Value
YES if the policy switch was successful, otherwise NO.

Discussion
Currently, NSApplicationActivationPolicyNone and NSApplicationActivationPolicyAccessory may be changed to NSApplicationActivationPolicyRegular, but other modifications are not supported.Needs links to running application

Availability
Available in Mac OS X v10.6 and later.

See Also
- activationPolicy (page 137)

Declared In
NSApplication.h

setApplicationIconImage:

Sets the receiver's icon to the specified image.

```
- (void)setApplicationIconImage:(NSImage *)anImage
```

Parameters
anImage
The image to use as the new application icon.

Discussion
This method sets the icon in the dock application tile. This method scales the image as necessary so that it fits in the dock tile. You can use this method to change your application icon while running. To restore your application's original icon, you pass `nil` to this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- applicationIconImage` (page 138)

Declared In
`NSApplication.h`

setDelegate:

Makes the given object the receiver's delegate.

```
- (void)setDelegate:(id < NSApplicationDelegate >)anObject
```

Parameters
anObject
The application delegate object.

Discussion
The messages a delegate can expect to receive are listed at the end of this specification. The delegate doesn't need to implement all the methods.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- delegate` (page 144)

Related Sample Code
CocoaDVDPlayer
PictureSharing

Declared In
`NSApplication.h`

setHelpMenu:

Sets the application's help menu.

```
- (void)setHelpMenu:(NSMenu *)helpMenu
```

Parameters
helpMenu
The menu to use as the application's help menu.

Discussion
If *helpMenu* is a non-nil menu it is set as the Help menu, and Spotlight for Help will be installed in it. If *helpMenu* is nil, AppKit will install Spotlight for Help into a menu of its choosing, and that menu is not returned from helpMenu.

If you wish to completely suppress Spotlight for Help, you can set a menu that does not appear in the menu bar.

NSApplication retains its Help menu and releases it when a different menu is set.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSApplication.h

setMainMenu:

Makes the given menu the receiver's main menu.

```
- (void)setMainMenu:(NSMenu *)aMenu
```

Parameters

aMenu

The new menu bar for the application.

Availability
Available in Mac OS X v10.0 and later.

See Also
- mainMenu (page 151)

Declared In
NSApplication.h

setPresentationOptions:

Sets the application presentation options to use when this application is active.

```
- (void)setPresentationOptions:(NSApplicationPresentationOptions)newOptions
```

Parameters

newOptions

The presentation options. The possible constants, and combination restrictions, are listed in "NSApplicationPresentationOptions" (page 187) and they combined using a C bitwise OR operator.

Discussion
Only certain combinations of "NSApplicationPresentationOptions" (page 187) flags are supported. When given an invalid combination of option flags this method raises an exception NSInvalidArgumentException exception..

Availability
Available in Mac OS X v10.6 and later.

See Also
- `currentSystemPresentationOptions` (page 143)
- `presentationOptions` (page 158)

Declared In
`NSApplication.h`

setServicesMenu:

Makes a given menu the receiver's Services menu.

```
- (void)setServicesMenu:(NSMenu *)aMenu
```

Parameters

aMenu

The new Services menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `servicesMenu` (page 167)

Declared In
`NSApplication.h`

setServicesProvider:

Registers a given object as the service provider.

```
- (void)setServicesProvider:(id)aProvider
```

Parameters

aProvider

The new service provider object.

Discussion
The service provider is an object that performs all services the application provides to other applications. When another application requests a service from the receiver, it sends the service request to *aProvider*. Service requests can arrive immediately after the service provider is set, so invoke this method only when your application is ready to receive requests.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `servicesProvider` (page 167)

Declared In
`NSApplication.h`

setWindowsMenu:

Makes the given menu the receiver's Window menu.

```
- (void)setWindowsMenu:(NSMenu *)aMenu
```

Parameters

aMenu

The new Window menu for the application.

Availability
Available in Mac OS X v10.0 and later.

See Also
- windowsMenu (page 181)

Declared In
NSApplication.h

setWindowsNeedUpdate:

Sets whether the receiver's windows need updating when the receiver has finished processing the current event.

```
- (void)setWindowsNeedUpdate:(BOOL)flag
```

Parameters

flag

If YES, the receiver's windows are updated after an event is processed.

Discussion
This method is especially useful for making sure menus are updated to reflect changes not initiated by user actions, such as messages received from remote objects.

Availability
Available in Mac OS X v10.0 and later.

See Also
- updateWindows (page 179)

Declared In
NSApplication.h

showHelp:

If your project is properly registered, and the necessary keys have been set in the property list, this method launches Help Viewer and displays the first page of your application's help book.

```
- (void)showHelp:(id)sender
```

Parameters

sender

The object that sent the command.

Discussion
For information on how to set up your project to take advantage of having Help Viewer display your help book, see Specifying the Comprehensive Help File.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `activateContextHelpMode:` (page 136)

Related Sample Code
HelpHook

Declared In
`NSHelpManager.h`

stop:

Stops the main event loop.

```
- (void)stop:(id)sender
```

Parameters

sender

The object that sent this message.

Discussion
This method notifies the application that you want to exit the current run loop as soon as it finishes processing the current `NSEvent` object. This method does not forcibly exit the current run loop. Instead it sets a flag that the application checks only after it finishes dispatching an actual event object. For example, you could call this method from an action method responding to a button click or from one of the many methods defined by the `NSResponder` class. However, calling this method from a timer or run-loop observer routine would not stop the run loop because they do not result in the p of an `NSEvent` object.

If you call this method from an event handler running in your main run loop, the application object exits out of the `run` method, thereby returning control to the `main()` function. If you call this method from within a modal event loop, it will exit the modal loop instead of the main event loop.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `run` (page 162)
- `runModalForWindow:` (page 163)
- `runModalSession:` (page 164)
- `terminate:` (page 176)

Declared In
`NSApplication.h`

stopModal

Stops a modal event loop.

```
- (void)stopModal
```

Discussion
This method should always be paired with a previous invocation of `runModalForWindow:` (page 163) or `beginModalSessionForWindow:` (page 139). When `runModalForWindow:` (page 163) is stopped with this method, it returns `NSRunStoppedResponse`. This method stops the loop only if it's executed by code responding to an event. If you need to stop a `runModalForWindow:` (page 163) loop outside of one of its event callbacks (for example, a method repeatedly invoked by an `NSTimer` object or a method running in a different thread), use the `abortModal` (page 135) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `runModalSession:` (page 164)
- `stopModalWithCode:` (page 174)

Related Sample Code
WhackedTV

Declared In
`NSApplication.h`

stopModalWithCode:

Stops a modal event loop, allowing you to return a custom result code.

```
- (void)stopModalWithCode:(NSInteger)returnCode
```

Parameters

returnCode
The result code you want returned from the `runModalForWindow:` or `runModalSession:` method. The meaning of this result code is up to you.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `abortModal` (page 135)
- `runModalForWindow:` (page 163)

Declared In
`NSApplication.h`

targetForAction:

Returns the object that receives the action message specified by the given selector

```
- (id)targetForAction:(SEL)aSelector
```

Parameters

aSelector

The desired action message.

Return Value

The object that would receive the specified action message or `nil` if no target object would receive the message. This method also returns `nil` if *aSelector* is `nil`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `sendAction:to:from:` (page 166)
- `tryToPerform:with:` (page 177)
- `targetForAction:to:from:` (page 175)

Declared In

`NSApplication.h`

targetForAction:to:from:

Finds an object that can receive the message specified by the given selector.

```
- (id)targetForAction:(SEL)anAction to:(id)aTarget from:(id)sender
```

Parameters

anAction

The desired action message.

aTarget

The first target object to check. Specify `nil` if you want the application to search the responder chain.

sender

The parameter to send to the action message.

Return Value

The object that can accept the specified action message or `nil` if no target object can receive the message. This method also returns `nil` if *anAction* is `nil`.

Discussion

If *aTarget* is not `nil`, *aTarget* is returned. If *aTarget* is `nil`, `NSApp` looks for an object that can respond to the message—that is, an object that implements a method matching *anAction*. The search begins with the first responder of the key window. If the first responder does not handle the message, it tries the first responder's next responder and continues following next responder links up the responder chain. If none of the objects in the key window's responder chain can handle the message, `NSApp` asks the key window's delegate whether it can handle the message.

If the delegate cannot handle the message and the main window is different from the key window, `NSApp` begins searching again with the first responder in the main window. If objects in the main window cannot handle the message, `NSApp` tries the main window's delegate. If it cannot handle the message, `NSApp` asks itself. If `NSApp` doesn't handle the message, it asks the application delegate. If there is no object capable of handling the message, `nil` is returned.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `sendAction:to:from:` (page 166)
- `tryToPerform:with:` (page 177)
- `targetForAction:` (page 174)

Declared In
`NSApplication.h`

terminate:

Terminates the receiver.

```
- (void)terminate:(id)sender
```

Parameters

sender

Typically, this parameter contains the object that initiated the termination request.

Discussion
This method is typically invoked when the user chooses Quit or Exit from the application's menu.

When invoked, this method performs several steps to process the termination request. First, it asks the application's document controller (if one exists) to save any unsaved changes in its documents. During this process, the document controller can cancel termination in response to input from the user. If the document controller does not cancel the operation, this method then calls the delegate's `applicationShouldTerminate:` (page 3576) method. If `applicationShouldTerminate:` (page 3576) returns NSTerminateCancel, the termination process is aborted and control is handed back to the main event loop. If the method returns `NSTerminateLater`, the application runs its run loop in the `NSModalPanelRunLoopMode` mode until the `replyToApplicationShouldTerminate:` method is called with the value `YES` or `NO`. If the `applicationShouldTerminate:` (page 3576) method returns NSTerminateNow, this method posts a `NSApplicationWillTerminateNotification` notification to the default notification center.

Do not bother to put final cleanup code in your application's `main()` function—it will never be executed. If cleanup is necessary, perform that cleanup in the delegate's `applicationWillTerminate:` (page 3579) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `run` (page 162)
- `stop:` (page 173)
- `applicationShouldTerminate:` (page 3576) (NSApplicationDelegate)
- `applicationWillTerminate:` (page 3579) (NSApplicationDelegate)
- `replyToApplicationShouldTerminate:` (page 160)

`NSApplicationWillTerminateNotification` (page 196)

Related Sample Code
PreLoginAgents
SimpleStickies
StickiesWithCoreData

WhackedTV

Declared In
NSApplication.h

tryToPerform:with:

Dispatches an action message to the specified target.

```
- (BOOL)tryToPerform:(SEL)aSelector with:(id)anObject
```

Parameters

aSelector
The action message you want to dispatch.

anObject
The target object that defines the specified selector.

Return Value
YES if either the receiver or its delegate can accept the specified selector; otherwise, NO. This method also returns NO if *aSelector* is nil.

Discussion
The receiver tries to perform the method *aSelector* using its inherited tryToPerform:with: (page 2202) method of NSResponder. If the receiver doesn't perform *aSelector*, the delegate is given the opportunity to perform it using its inherited performSelector:withObject: method of NSObject.

Availability
Available in Mac OS X v10.0 and later.

See Also
respondsToSelector: (NSObject protocol)

Declared In
NSApplication.h

unhide:

Restores hidden windows to the screen and makes the receiver active.

```
- (void)unhide:(id)sender
```

Parameters

sender
The object that sent the command.

Discussion
Invokes unhideWithoutActivation (page 178).

Availability
Available in Mac OS X v10.0 and later.

See Also
- activateIgnoringOtherApps: (page 136)

- hide: (page 148)

Declared In
NSApplication.h

unhideAllApplications:

Unhides all applications, including the receiver.

- (void)unhideAllApplications:(id)*sender*

Parameters

sender

The object that sent this message.

Discussion
This action causes each application to order its windows to the front, which could obscure the currently active window in the active application.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSApplication.h

unhideWithoutActivation

Restores hidden windows without activating their owner (the receiver).

- (void)unhideWithoutActivation

Discussion
When this method begins, it posts an NSApplicationWillUnhideNotification (page 196) to the default notification center. If it completes successfully, it posts an NSApplicationDidUnhideNotification (page 195).

Availability
Available in Mac OS X v10.0 and later.

See Also
- activateIgnoringOtherApps: (page 136)
- hide: (page 148)
- applicationDidUnhide: (page 3573) (NSApplicationDelegate)
- applicationWillUnhide: (page 3579) (NSApplicationDelegate)

Declared In
NSApplication.h

unregisterUserInterfaceItemSearchHandler:

Unregister an an object that provides help data to your application.

```
- (void)unregisterUserInterfaceItemSearchHandler:(id < NSUserInterfaceItemSearching
    >)handler
```

Parameters

handler

The class instance that conforms to `NSUserInterfaceItemSearching` and provides help content.

Discussion

If you unregister the same instance more than once the subsequent invocations are ignored. Unregistering an instance that was never registered is ignored.

Availability

Available in Mac OS X v10.6 and later.

See Also

- `registerUserInterfaceItemSearchHandler:` (page 159)

Declared In

`NSUserInterfaceItemSearching.h`

updateWindows

Sends an `update` (page 3405) message to each onscreen window.

```
- (void)updateWindows
```

Discussion

This method is invoked automatically in the main event loop after each event when running in `NSDefaultRunLoopMode` or `NSModalRunLoopMode`. This method is not invoked automatically when running in `NSEventTrackingRunLoopMode`.

When this method begins, it posts an `NSApplicationWillUpdateNotification` (page 196) to the default notification center. When it successfully completes, it posts an `NSApplicationDidUpdateNotification` (page 195).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `update` (page 3405) (NSWindow)
- `setWindowsNeedUpdate:` (page 172)
- `applicationDidUpdate:` (page 3573) (NSApplicationDelegate)
- `applicationWillUpdate:` (page 3580) (NSApplicationDelegate)

Declared In

`NSApplication.h`

updateWindowsItem:

Updates the Window menu item for a given window to reflect the edited status of that window.

```
- (void)updateWindowsItem:(NSWindow *)aWindow
```

Parameters

aWindow

The window whose menu item is to be updated.

Discussion

You rarely need to invoke this method because it is invoked automatically when the edit status of an `NSWindow` object is set.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `changeWindowsItem:title:filename:` (page 141)
- `setDocumentEdited:` (page 3379) (`NSWindow`)

Declared In

`NSApplication.h`

userInterfaceLayoutDirection

Returns the layout direction of the user interface.

```
- (NSUserInterfaceLayoutDirection)userInterfaceLayoutDirection
```

Return Value

The direction of the user interface layout. See "NSUserInterfaceLayoutDirection" (page 184) for possible values.

Discussion

This method specifies the general user interface layout flow directions.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSApplication.h`

validRequestorForSendType:returnType:

Indicates whether the receiver can send and receive the specified pasteboard types.

```
- (id)validRequestorForSendType:(NSString *)sendType returnType:(NSString
    *)returnType
```

Parameters

sendType

The pasteboard type the application needs to send.

returnType

The pasteboard type the application needs to receive.

Return Value

The object that can send and receive the specified types or `nil` if the receiver knows of no object that can send and receive data of that type.

Discussion
This message is sent to all responders in a responder chain. `NSApp` is typically the last item in the responder chain, so it usually receives this message only if none of the current responders can send *`sendType`* data and accept back *`returnType`* data.

The receiver passes this message on to its delegate if the delegate can respond (and isn't an `NSResponder` object with its own next responder). If the delegate cannot respond or returns `nil`, this method returns `nil`. If the delegate can find an object that can send *`sendType`* data and accept back *`returnType`* data, it returns that object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `registerServicesMenuSendTypes:returnTypes:` (page 158)
- `validRequestorForSendType:returnType:` (page 2204) (NSResponder)
- `readSelectionFromPasteboard:` (page 3797) (NSServicesRequests protocol)
- `writeSelectionToPasteboard:types:` (page 3798) (NSServicesRequests protocol)

Declared In
`NSApplication.h`

windows

Returns an array containing the receiver's window objects.

```
- (NSArray *)windows
```

Return Value
An array of `NSWindow` objects. This array includes both onscreen and offscreen windows.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
GLUT
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSApplication.h`

windowsMenu

Returns the Window menu of the application.

```
- (NSMenu *)windowsMenu
```

Return Value
The window menu or `nil` if such a menu does not exist or has not yet been created.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setWindowsMenu: (page 172)

Declared In
NSApplication.h

windowWithWindowNumber:

Returns the window corresponding to the specified window number.

```
- (NSWindow *)windowWithWindowNumber:(NSInteger)windowNum
```

Parameters

windowNum
The unique window number associated with the desired NSWindow object.

Return Value
The desired window object or nil if the window could not be found.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSApplication.h

Delegate Methods

application:delegateHandlesKey:

Sent by Cocoa's built-in scripting support during execution of get or set script commands to find out if the delegate can handle operations on the specified key-value key.

```
- (BOOL)application:(NSApplication *)sender delegateHandlesKey:(NSString *)key
```

Parameters

sender
The application object associated with the delegate.

key
The key to be handled.

Return Value
YES if your delegate handles the key or NO if it does not.

Discussion
The method should return YES if the delegate for the application *sender* handles the key specified by *key*, which means it can get or set the scriptable property or element that corresponds to that key. The application implements methods for each of the keys that it handles, where the method name matches the key.

For example, a scriptable application that doesn't use Cocoa's document-based application architecture can implement this method to supply its own document ordering. Such an application might want to do this because the standard application delegate expects to work with a document-based application. The TextEdit application (whose source is distributed with Mac OS X developer tools) provides the following implementation:

```
return [key isEqualToString:@"orderedDocuments"];
```

TextEdit then implements the `orderedDocuments` method in its controller class to return an ordered list of documents. An application with its own window ordering might add a test for the key `orderedWindows` so that its delegate can provide its own version of `orderedWindows`.

Important: Cocoa scripting does not invoke this method for script commands other than `get` or `set`. For information on working with other commands, see Script Commands in *Cocoa Scripting Guide*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `orderedDocuments` (page 154) (NSApplication)
- `orderedWindows` (page 155) (NSApplication)

Declared In
`NSApplicationScripting.h`

application:printFiles:

(Available in Mac OS X v10.3 through Mac OS X v10.5. Use `application:printFiles:withSettings:showPrintPanels:` (page 3569) instead.)

```
- (void)application:(NSApplication *)sender
    printFiles:(NSArray *)filenames
```

Discussion
Identical to `application:printFile:` (page 3568) except that the receiver prints multiple files corresponding to the file names in the *`filenames`* array.

Delegates should invoke the `replyToOpenOrPrint:` (page 161) method upon success or failure, or when the user cancels the operation.

Availability
Available in Mac OS X v10.3 through Mac OS X v10.5.

Declared In
`NSApplication.h`

Constants

NSUserInterfaceLayoutDirection

Specifies the directional flow of the user interface. These constants are returned by `userInterfaceLayoutDirection` (page 180).

```
enum {
    NSUserInterfaceLayoutDirectionLeftToRight = 0,
    NSUserInterfaceLayoutDirectionRightToLeft = 1
};
typedef NSInteger NSUserInterfaceLayoutDirection;
```

Constants

`NSUserInterfaceLayoutDirectionLeftToRight`

Layout direction is left to right.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSUserInterfaceLayoutDirectionRightToLeft`

Layout direction is right to left. This is appropriate when running with localizations such as Arabic or Hebrew that should have the user interface layout origin on the right edge of the coordinate system.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

Return values for modal operations

These are possible return values for `runModalForWindow:` (page 163) and `runModalSession:` (page 164).

```
enum {
    NSRunStoppedResponse    = (-1000),
    NSRunAbortedResponse    = (-1001),
    NSRunContinuesResponse  = (-1002)
};
```

Constants

`NSRunStoppedResponse`

Modal session was broken with `stopModal` (page 174).

Available in Mac OS X v10.0 and later.

Declared in `NSApplication.h`.

`NSRunAbortedResponse`

Modal session was broken with `abortModal` (page 135).

Available in Mac OS X v10.0 and later.

Declared in `NSApplication.h`.

`NSRunContinuesResponse`

Modal session is continuing (returned by `runModalSession:` (page 164) only).

Available in Mac OS X v10.0 and later.

Declared in `NSApplication.h`.

Discussion
The system also reserves all values below these.

NSUpdateWindowsRunLoopOrdering

This constant is used by the `NSRunLoop` method `performSelector:target:argument:order:modes:`.

```
enum {
    NSUpdateWindowsRunLoopOrdering = 500000
};
```

Constants
`NSUpdateWindowsRunLoopOrdering`
Run-loop message priority for handling window updates.

Available in Mac OS X v10.0 and later.

Declared in `NSApplication.h`.

NSApp

A global constant for the shared application instance.

```
id NSApp
```

Constants
`NSApp`
Global constant for the shared application instance. This is the same as sending the `NSApplication` class the method `sharedApplication` (page 135) message.

Available in Mac OS X v10.0 and later.

Declared in `NSApplication.h`.

Discussion
This variable designates the shared application object, created by the sharedApplication (page 135) method.

Declared In
`NSApplication.h`

NSRequestUserAttentionType

These constants specify the level of severity of a user attention request and are used by `cancelUserAttentionRequest:` (page 141) and `requestUserAttention:` (page 161).

```
enum {
    NSCriticalRequest = 0,
    NSInformationalRequest = 10
}
typedef NSUInteger NSRequestUserAttentionType;
```

Constants

`NSCriticalRequest`

The user attention request is a critical request.

The dock icon will bounce until either the application becomes active or the request is canceled.

Available in Mac OS X v10.1 and later.

Declared in `NSApplication.h`.

`NSInformationalRequest`

The user attention request is an informational request.

The dock icon will bounce for one second. The request, though, remains active until either the application becomes active or the request is canceled.

Available in Mac OS X v10.1 and later.

Declared in `NSApplication.h`.

Declared In

`NSApplication.h`

NSApplicationDelegateReply

These constants indicate whether or not a copy or print operation was successful, was cancelled, or failed. These constants are used by the `replyToOpenOrPrint:` (page 161) method.

```
enum NSApplicationDelegateReply {
    NSApplicationDelegateReplySuccess = 0,
    NSApplicationDelegateReplyCancel = 1,
    NSApplicationDelegateReplyFailure = 2
}
typedef NSUInteger NSApplicationDelegateReply;
```

Constants

`NSApplicationDelegateReplySuccess`

Indicates the operation succeeded.

Available in Mac OS X v10.3 and later.

Declared in `NSApplication.h`.

`NSApplicationDelegateReplyCancel`

Indicates the user cancelled the operation.

Available in Mac OS X v10.3 and later.

Declared in `NSApplication.h`.

`NSApplicationDelegateReplyFailure`

Indicates an error occurred processing the operation.

Available in Mac OS X v10.3 and later.

Declared in `NSApplication.h`.

NSApplicationPresentationOptions

The constants control the presentation of the application. Typically, in fullscreen applications such as games or kiosks. They are used by the methods `presentationOptions` (page 158), `currentSystemPresentationOptions` (page 143), and `setPresentationOptions` (page 170).

```
enum {
    NSApplicationPresentationDefault                    = 0,
    NSApplicationPresentationAutoHideDock               = (1 <<  0),
    NSApplicationPresentationHideDock                   = (1 <<  1),
    NSApplicationPresentationAutoHideMenuBar            = (1 <<  2),
    NSApplicationPresentationHideMenuBar                = (1 <<  3),
    NSApplicationPresentationDisableAppleMenu           = (1 <<  4),
    NSApplicationPresentationDisableProcessSwitching    = (1 <<  5),
    NSApplicationPresentationDisableForceQuit           = (1 <<  6),
    NSApplicationPresentationDisableSessionTermination  = (1 <<  7),
    NSApplicationPresentationDisableHideApplication     = (1 <<  8),
    NSApplicationPresentationDisableMenuBarTransparency = (1 <<  9)
};
typedef NSUInteger NSApplicationPresentationOptions;
```

Constants

`NSApplicationPresentationDefault`

This is the default presentation mode. The application is displayed as normal, including dock, menu bar, process switching is enabled, force quit is enabled, session termination is enabled, the hide menu is enabled, and the menu bar transparency is normal.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSApplicationPresentationAutoHideDock`

The dock is normally hidden, but automatically appears when moused near.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSApplicationPresentationHideDock`

The dock is entirely hidden and disabled.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSApplicationPresentationAutoHideMenuBar`

The menu bar is normally hidden, but automatically appears when moused near.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSApplicationPresentationHideMenuBar`

The menu bar is entirely hidden and disabled.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSApplicationPresentationDisableAppleMenu`

All Apple Menu items are disabled.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSApplicationPresentationDisableProcessSwitching`
The process switching user interface (Command + Tab to cycle through applications) is disabled.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSApplicationPresentationDisableForceQuit`
The force quit panel (displayed by pressing Command + Option + Esc) is disabled

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSApplicationPresentationDisableSessionTermination`
The panel that shows the Restart, Shut Down, and Log Out options that are displayed as a result of pushing the power key is disabled.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSApplicationPresentationDisableHideApplication`
The application's "Hide" menu item is disabled.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSApplicationPresentationDisableMenuBarTransparency`
The menu bar transparency appearance is disabled.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

Discussion
There are restrictions on the combination of presentation options that can be set simultaneously:

- `NSApplicationPresentationAutoHideDock` (page 187) and `NSApplicationPresentationHideDock` (page 187) are mutually exclusive: You may specify one or the other, but not both.
- `NSApplicationPresentationAutoHideMenuBar` (page 187) and `NSApplicationPresentationHideMenuBar` (page 187) are mutually exclusive: You may specify one or the other, but not both.
- If you specify `NSApplicationPresentationHideMenuBar` (page 187), it must be accompanied by `NSApplicationPresentationHideDock` (page 187).
- If you specify `NSApplicationPresentationAutoHideMenuBar` (page 187), it must be accompanied by either `NSApplicationPresentationHideDock` (page 187) or `NSApplicationPresentationAutoHideDock` (page 187).
- If you specify any of `NSApplicationPresentationDisableProcessSwitching` (page 188), `NSApplicationPresentationDisableForceQuit` (page 188), `NSApplicationPresentationDisableSessionTermination` (page 188), or `NSApplicationPresentationDisableMenuBarTransparency` (page 188), it must be accompanied by either `NSApplicationPresentationHideDock` (page 187) or `NSApplicationPresentationAutoHideDock` (page 187).

When `setPresentationOptions:` (page 170) receives a parameter value that does not conform to these requirements, it raises an `NSInvalidArgumentException`.

NSApplicationTerminateReply

These constants define whether an application should terminate and are used by the delegate method `applicationShouldTerminate:` (page 3576).

```
enum {
    NSTerminateCancel = 0,
    NSTerminateNow    = 1,
    NSTerminateLater  = 2
}
typedef NSUInteger NSApplicationTerminateReply;
```

Constants

`NSTerminateNow`

It is OK to proceed with termination.

Available in Mac OS X v10.0 and later.

Declared in `NSApplication.h`.

`NSTerminateCancel`

The application should not be terminated.

Available in Mac OS X v10.0 and later.

Declared in `NSApplication.h`.

`NSTerminateLater`

It may be OK to proceed with termination later. Returning this value causes Cocoa to run the run loop in the `NSModalPanelRunLoopMode` until your application subsequently calls `replyToApplicationShouldTerminate:` (page 160) with the value `YES` or `NO`. This return value is for delegates that need to provide document modal alerts (sheets) in order to decide whether to quit.

Available in Mac OS X v10.0 and later.

Declared in `NSApplication.h`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSApplication.h`

NSApplicationPrintReply

These constants are returned by the delegate method `application:printFiles:withSettings:showPrintPanels:` (page 3569).

```
enum {
    NSPrintingCancelled  = 0,
    NSPrintingSuccess    = 1,
    NSPrintingFailure    = 3,
    NSPrintingReplyLater = 2
}
typedef NSUInteger NSApplicationPrintReply;
```

Constants

`NSPrintingCancelled`

Printing was cancelled.

Available in Mac OS X v10.4 and later.

Declared in `NSApplication.h`.

`NSPrintingSuccess`

Printing was successful.

Available in Mac OS X v10.4 and later.

Declared in `NSApplication.h`.

`NSPrintingFailure`

Printing failed.

Available in Mac OS X v10.4 and later.

Declared in `NSApplication.h`.

`NSPrintingReplyLater`

The result of printing cannot be returned immediately, for example, if printing will cause the presentation of a sheet. If your method returns `NSPrintingReplyLater` it must always invoke `replyToOpenOrPrint:` (page 161) when the entire print operation has been completed, successfully or not.

Available in Mac OS X v10.4 and later.

Declared in `NSApplication.h`.

Declared In

`NSApplication.h`

Run loop modes

These loop mode constants are defined by `NSApplication`.

```
NSString *NSModalPanelRunLoopMode;
NSString *NSEventTrackingRunLoopMode;
```

Constants

`NSEventTrackingRunLoopMode`

A run loop should be set to this mode when tracking events modally, such as a mouse-dragging loop.

Available in Mac OS X v10.0 and later.

Declared in `NSApplication.h`.

`NSModalPanelRunLoopMode`

A run loop should be set to this mode when waiting for input from a modal panel, such as `NSSavePanel` or `NSOpenPanel`.

Available in Mac OS X v10.0 and later.

Declared in `NSApplication.h`.

Declared In
NSApplication.h

NSAppKitVersionNumber

This constant identifies the installed version of the Application Kit framework.

```
const double NSAppKitVersionNumber;
```

Constants

NSAppKitVersionNumber

This value corresponds to one of the constants defined in "Application Kit framework version numbers" (page 191).

Available in Mac OS X v10.1 and later.

Declared in NSApplication.h.

Application Kit framework version numbers

You can use the following constants to determine if you are using a version of the Application Kit framework newer than the version delivered in Mac OS X v10.0.

```
#define NSAppKitVersionNumber10_0   577
#define NSAppKitVersionNumber10_1   620
#define NSAppKitVersionNumber10_2   663
#define NSAppKitVersionNumber10_2_3 663.6
#define NSAppKitVersionNumber10_3   743
#define NSAppKitVersionNumber10_3_2 743.14
#define NSAppKitVersionNumber10_3_3 743.2
#define NSAppKitVersionNumber10_3_5 743.24
#define NSAppKitVersionNumber10_3_7 743.33
#define NSAppKitVersionNumber10_3_9 743.36
#define NSAppKitVersionNumber10_4   824
#define NSAppKitVersionNumber10_4_1 824.1
#define NSAppKitVersionNumber10_4_3 824.23
#define NSAppKitVersionNumber10_4_4 824.33
#define NSAppKitVersionNumber10_4_7 824.41
#define NSAppKitVersionNumber10_5   949
#define NSAppKitVersionNumber10_5_2 949.27
#define NSAppKitVersionNumber10_5_3 949.33
```

Constants

NSAppKitVersionNumber10_0

The Application Kit framework included in Mac OS X v10.0.

Available in Mac OS X v10.1 and later.

Declared in NSApplication.h.

NSAppKitVersionNumber10_1

The Application Kit framework included in Mac OS X v10.1.

Available in Mac OS X v10.2 and later.

Declared in NSApplication.h.

`NSAppKitVersionNumber10_2`
The Application Kit framework included in Mac OS X v10.2.
Available in Mac OS X v10.3 and later.
Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_2_3`
The Application Kit framework included in Mac OS X v10.2.3.
Available in Mac OS X v10.3 and later.
Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_3`
The Application Kit framework included in Mac OS X v10.3.
Available in Mac OS X v10.4 and later.
Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_3_2`
The Application Kit framework included in Mac OS X v10.3.2.
Available in Mac OS X v10.4 and later.
Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_3_3`
The Application Kit framework included in Mac OS X v10.3.3.
Available in Mac OS X v10.4 and later.
Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_3_5`
The Application Kit framework included in Mac OS X v10.3.5.
Available in Mac OS X v10.4 and later.
Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_3_7`
The Application Kit framework included in Mac OS X v10.3.7.
Available in Mac OS X v10.5 and later.
Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_3_9`
The Application Kit framework included in Mac OS X v10.3.9.
Available in Mac OS X v10.5 and later.
Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_4`
The Application Kit framework included in Mac OS X v10.4.
Available in Mac OS X v10.5 and later.
Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_4_1`
The Application Kit framework included in Mac OS X v10.4.1.
Available in Mac OS X v10.6 and later.
Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_4_3`
The Application Kit framework included in Mac OS X v10.4.3.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_4_4`
The Application Kit framework included in Mac OS X v10.4.4.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_4_7`
The Application Kit framework included in Mac OS X v10.4.7.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_5`
The Application Kit framework included in Mac OS X v10.5.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_5_2`
The Application Kit framework included in Mac OS X v10.5.2.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

`NSAppKitVersionNumber10_5_3`
The Application Kit framework included in Mac OS X v10.5.3.

Available in Mac OS X v10.6 and later.

Declared in `NSApplication.h`.

Notifications

These notifications apply to `NSApplication`. See `Notifications` in *NSWorkspace Class Reference* for additional, similar notifications.

NSApplicationDidBecomeActiveNotification

Posted immediately after the application becomes active.

The notification object is `sharedApplication` (page 135). This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationDidChangeScreenParametersNotification

Posted when the configuration of the displays attached to the computer is changed.

The configuration change can be made either programmatically or when the user changes settings in the Displays control panel. The notification object is `sharedApplication` (page 135). This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationDidFinishLaunchingNotification

Posted at the end of the `finishLaunching` (page 147) method to indicate that the application has completed launching and is ready to run.

The notification object is `sharedApplication` (page 135). This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationDidHideNotification

Posted at the end of the `hide:` (page 148) method to indicate that the application is now hidden.

The notification object is `NSApp`. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationDidResignActiveNotification

Posted immediately after the application gives up its active status to another application.

The notification object is `NSApp`. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationDidUnhideNotification

Posted at the end of the `unhideWithoutActivation` (page 178) method to indicate that the application is now visible.

The notification object is `NSApp`. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationDidUpdateNotification

Posted at the end of the `updateWindows` (page 179) method to indicate that the application has finished updating its windows.

The notification object is `NSApp`. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationWillBecomeActiveNotification

Posted immediately after the application becomes active.

The notification object is `NSApp`. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationWillFinishLaunchingNotification

Posted at the start of the `finishLaunching` (page 147) method to indicate that the application has completed its initialization process and is about to finish launching.

The notification object is `NSApp`. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationWillHideNotification

Posted at the start of the `hide:` (page 148) method to indicate that the application is about to be hidden.

The notification object is `NSApp`. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationWillResignActiveNotification

Posted immediately before the application gives up its active status to another application.

The notification object is `sharedApplication` (page 135). This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationWillTerminateNotification

Posted by the `terminate:` (page 176) method to indicate that the application will terminate.

Posted only if the delegate method `applicationShouldTerminate:` (page 3576) returns `YES`. The notification object is `sharedApplication` (page 135). This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationWillUnhideNotification

Posted at the start of the `unhideWithoutActivation` (page 178) method to indicate that the application is about to become visible.

The notification object is `sharedApplication` (page 135). This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSApplicationWillUpdateNotification

Posted at the start of the `updateWindows` (page 179) method to indicate that the application is about to update its windows.

The notification object is `sharedApplication` (page 135). This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSArrayController Class Reference

Inherits from	NSObjectController : NSController : NSObject
Conforms to	NSCoding (NSController) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.3 and later.
Declared in	AppKit/NSArrayController.h
Companion guides	Cocoa Bindings Programming Topics Predicate Programming Guide Core Data Programming Guide
Related sample code	CoreRecipes DemoMonkey GridCalendar LightTable XMLBrowser

Overview

`NSArrayController` is a bindings compatible class that manages a collection of objects. Typically the collection is an array, however, if the controller manages a relationship of a managed object (see `NSManagedObject`) the collection may be a set. `NSArrayController` provides selection management and sorting capabilities.

Tasks

Managing Sort Descriptors

`- setSortDescriptors:` (page 224)
Sets the sort descriptors for the receiver.

`- sortDescriptors` (page 224)
Returns the receiver's array of sort descriptors.

Arranging Objects

- `arrangeObjects:` (page 206)
  Returns a given array, appropriately sorted and filtered.
- `arrangedObjects` (page 205)
  Returns an array containing the receiver's content objects arranged using `arrangeObjects:` (page 206).
- `rearrangeObjects` (page 213)
  Triggers filtering of the receiver's content.

Managing Content

- `add:` (page 203)
  Creates and adds a new object to the receiver's content and arranged objects.
- `setAutomaticallyPreparesContent:` (page 219)
  Sets whether the receiver automatically creates and inserts new content objects automatically.
- `automaticallyPreparesContent` (page 206)
  Returns a Boolean value that indicates whether the receiver automatically prepares its content when it is loaded from a nib.

Selection Attributes

- `setAvoidsEmptySelection:` (page 220)
  Sets whether the receiver attempts to avoid an empty selection.
- `avoidsEmptySelection` (page 208)
  Returns a Boolean value that indicates whether if the receiver requires that the content array attempt to maintain a selection.
- `setClearsFilterPredicateOnInsertion:` (page 221)
  Sets whether the receiver automatically clears an existing filter predicate when a new object is inserted or added to the content array.
- `preservesSelection` (page 212)
  Returns a Boolean value that indicates whether the receiver will attempt to preserve the current selection when the content changes.
- `setPreservesSelection:` (page 222)
  Sets whether the receiver attempts to preserve selection when the content changes.
- `alwaysUsesMultipleValuesMarker` (page 205)
  Returns a Boolean value that indicates whether the receiver always returns the multiple values marker when multiple objects are selected.
- `setAlwaysUsesMultipleValuesMarker:` (page 219)
  Sets whether the receiver always returns the multiple values marker when multiple objects are selected.

Managing Selections

- `selectionIndex` (page 217)
  Returns the index of the first object in the receiver's selection.
- `setSelectionIndex:` (page 222)
  Sets the receiver's selection to the given index, and returns a Boolean value that indicates whether the selection was changed.
- `setSelectsInsertedObjects:` (page 224)
  Sets whether the receiver will automatically select objects as they are inserted.
- `selectsInsertedObjects` (page 219)
  Returns whether the receiver automatically selects inserted objects.
- `setSelectionIndexes:` (page 223)
  Sets the receiver's selection indexes and returns a Boolean value that indicates whether the selection changed.
- `selectionIndexes` (page 217)
  Returns an index set containing the indexes of the receiver's currently selected objects in the content array.
- `addSelectionIndexes:` (page 205)
  Adds the objects at the specified indexes in the receiver's content array to the current selection, returning `YES` if the selection was changed.
- `removeSelectionIndexes:` (page 216)
  Removes the object as the specified *`indexes`* from the receiver's current selection, returning `YES` if the selection was changed.
- `setSelectedObjects:` (page 222)
  Sets *`objects`* as the receiver's current selection, returning `YES` if the selection was changed.
- `selectedObjects` (page 216)
  Returns an array containing the receiver's selected objects.
- `addSelectedObjects:` (page 204)
  Adds *`objects`* from the receiver's content array to the current selection, returning `YES` if the selection was changed.
- `removeSelectedObjects:` (page 215)
  Removes *`objects`* from the receiver's current selection, returning `YES` if the selection was changed.
- `selectNext:` (page 218)
  Selects the next object, relative to the current selection, in the receiver's arranged content.
- `canSelectNext` (page 209)
  Returns `YES` if the next object, relative to the current selection, in the receiver's content array can be selected.
- `selectPrevious:` (page 218)
  Selects the previous object, relative to the current selection, in the receiver's arranged content.
- `canSelectPrevious` (page 209)
  Returns `YES` if the previous object, relative to the current selection, in the receiver's content array can be selected.

Inserting

- `canInsert` (page 208)
  Returns a Boolean value that indicates whether an object can be inserted into the receiver's content collection.
- `insert:` (page 211)
  Creates a new object and inserts it into the receiver's content array.

Adding and Removing Objects

- `addObject:` (page 203)
  Adds *object* to the receiver's content collection and the arranged objects array.
- `addObjects:` (page 204)
  Adds *objects* to the receiver's content collection.
- `insertObject:atArrangedObjectIndex:` (page 211)
  Inserts *object* into the receiver's arranged objects array at the location specified by *index*, and adds it to the receiver's content collection.
- `insertObjects:atArrangedObjectIndexes:` (page 212)
  Inserts *objects* into the receiver's arranged objects array at the locations specified in *indexes*, and adds it to the receiver's content collection.
- `removeObjectAtArrangedObjectIndex:` (page 214)
  Removes the object at the specified *index* in the receiver's arranged objects from the receiver's content array.
- `removeObjectsAtArrangedObjectIndexes:` (page 215)
  Removes the objects at the specified *indexes* in the receiver's arranged objects from the content array.
- `remove:` (page 213)
  Removes the receiver's selected objects from the content collection.
- `removeObject:` (page 214)
  Removes *object* from the receiver's content collection.
- `removeObjects:` (page 215)
  Removes *objects* from the receiver's content collection.

Filtering Content

- `clearsFilterPredicateOnInsertion` (page 209)
  Returns a Boolean value that indicates whether the receiver automatically clears an existing filter predicate when new items are inserted or added to the content.
- `filterPredicate` (page 210)
  Returns the predicate used by the receiver to filter the array controller contents.
- `setFilterPredicate:` (page 221)
  Sets the predicate used to filter the contents of the receiver.

Automatic Content Rearranging

- `setAutomaticallyRearrangesObjects:` (page 220)
  Sets whether or not the receiver automatically rearranges its content to correspond to the current sort descriptors and filter predicates.
- `automaticallyRearrangesObjects` (page 207)
  Returns a Boolean that indicates if the receiver automatically rearranges its content to correspond to the current sort descriptors and filter predicates.
- `automaticRearrangementKeyPaths` (page 207)
  Returns an array of key paths that trigger automatic content sorting or filtering.
- `didChangeArrangementCriteria` (page 210)
  Invoked when any criteria for arranging objects change.

Instance Methods

add:

Creates and adds a new object to the receiver's content and arranged objects.

```
- (void)add:(id)sender
```

Parameters

sender
  Typically the object that invoked this method.

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism (see Error Responders and Error Recovery) can provide feedback as a sheet.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSArrayController.h`

addObject:

Adds *object* to the receiver's content collection and the arranged objects array.

```
- (void)addObject:(id)object
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `addObjects:` (page 204)
- `insertObject:atArrangedObjectIndex:` (page 211)

- removeObject: (page 214)

Related Sample Code
Aperture Edit Plugin - Borders & Titles
QTCompressionOptionsWindow
XMLBrowser

Declared In
NSArrayController.h

addObjects:

Adds *objects* to the receiver's content collection.

```
- (void)addObjects:(NSArray *)objects
```

Discussion
If selectsInsertedObjects (page 219) returns YES (the default), the added objects are selected in the array controller.

It is important to note that inserting many objects with selectsInsertedObjects on can cause a significant performance penalty. In this case it is more efficient to use the setContent: (page 1756) method to set the array, or to set selectsInsertedObjects to NO before adding the objects with addObjects:.

Availability
Available in Mac OS X v10.3 and later.

See Also
- addObject: (page 203)
- insertObjects:atArrangedObjectIndexes: (page 212)
- removeObjects: (page 215)

Declared In
NSArrayController.h

addSelectedObjects:

Adds *objects* from the receiver's content array to the current selection, returning YES if the selection was changed.

```
- (BOOL)addSelectedObjects:(NSArray *)objects
```

Discussion
Attempting to change the selection may cause a commitEditing (page 850) message which fails, thus denying the selection change.

Availability
Available in Mac OS X v10.3 and later.

See Also
- removeSelectedObjects: (page 215)
- setSelectedObjects: (page 222)

Declared In
NSArrayController.h

addSelectionIndexes:

Adds the objects at the specified indexes in the receiver's content array to the current selection, returning YES if the selection was changed.

```
- (BOOL)addSelectionIndexes:(NSIndexSet *)indexes
```

Discussion
Attempting to change the selection may cause a commitEditing (page 850) message which fails, thus denying the selection change.

Availability
Available in Mac OS X v10.3 and later.

See Also
- removeSelectionIndexes: (page 216)

Declared In
NSArrayController.h

alwaysUsesMultipleValuesMarker

Returns a Boolean value that indicates whether the receiver always returns the multiple values marker when multiple objects are selected.

```
- (BOOL)alwaysUsesMultipleValuesMarker
```

Return Value
YES if the receiver always returns the multiple values marker when multiple objects are selected—even if the selected items have the same value, otherwise NO.

Discussion
The default is NO.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setAlwaysUsesMultipleValuesMarker: (page 219)

Declared In
NSArrayController.h

arrangedObjects

Returns an array containing the receiver's content objects arranged using arrangeObjects: (page 206).

```
- (id)arrangedObjects
```

Return Value
An array containing the receiver's content objects arranged using `arrangeObjects:` (page 206).

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `arrangeObjects:` (page 206)

Related Sample Code
CoreRecipes
DemoMonkey
QTCompressionOptionsWindow
With and Without Bindings

Declared In
`NSArrayController.h`

arrangeObjects:

Returns a given array, appropriately sorted and filtered.

```
- (NSArray *)arrangeObjects:(NSArray *)objects
```

Return Value
An array containing *objects* filtered using the receiver's filter predicate (see filterPredicate (page 210)) and sorted according to the receiver's `sortDescriptors` (page 224).

Discussion
Subclasses should override this method to use a different sort mechanism, provide custom object arrangement, or (typically only prior to Mac OS X version 10.4, which provides a filter predicate) filter the objects.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `arrangedObjects` (page 205)
- `rearrangeObjects` (page 213)
- `sortDescriptors` (page 224)

Declared In
`NSArrayController.h`

automaticallyPreparesContent

Returns a Boolean value that indicates whether the receiver automatically prepares its content when it is loaded from a nib.

```
- (BOOL)automaticallyPreparesContent
```

Return Value
`YES` if the receiver automatically prepares its content when loaded from a nib, otherwise `NO`.

Discussion
See `setAutomaticallyPreparesContent:` (page 219) for a full explanation of "automatically prepares content."

The default is `YES`.

See Also
- `setAutomaticallyPreparesContent:` (page 219)
- `prepareContent` (page 1753)

automaticallyRearrangesObjects

Returns a Boolean that indicates if the receiver automatically rearranges its content to correspond to the current sort descriptors and filter predicates.

```
- (BOOL)automaticallyRearrangesObjects
```

Return Value
`YES` if the receiver automatically rearranges objects upon changes to the content, `NO` if the content does not automatically rearrange.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSArrayController.h`

automaticRearrangementKeyPaths

Returns an array of key paths that trigger automatic content sorting or filtering.

```
- (NSArray *)automaticRearrangementKeyPaths
```

Return Value
An array of key paths that trigger automatic content sorting or filtering.

Discussion
Subclasses can override this method to customize the default behavior of the sort descriptors and filtering predicates, for example, if additional arrangement criteria are used in a custom implementation of `rearrangeObjects` (page 213).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `rearrangeObjects` (page 213)

Declared In
NSArrayController.h

avoidsEmptySelection

Returns a Boolean value that indicates whether if the receiver requires that the content array attempt to maintain a selection.

```
- (BOOL)avoidsEmptySelection
```

Return Value
YES if the receiver requires that the content array attempt to maintain a selection at all times, otherwise NO.

Discussion
The default is YES.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setAvoidsEmptySelection: (page 220)

Declared In
NSArrayController.h

canInsert

Returns a Boolean value that indicates whether an object can be inserted into the receiver's content collection.

```
- (BOOL)canInsert
```

Return Value
YES if an object can be inserted into the receiver's content collection, otherwise NO.

Discussion
The result of this method can be used by a binding to enable user interface items.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- insert: (page 211)

Declared In
NSArrayController.h

canSelectNext

Returns `YES` if the next object, relative to the current selection, in the receiver's content array can be selected.

```
- (BOOL)canSelectNext
```

Discussion
The result of this method can be used by a binding to enable user interface items.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `selectNext:` (page 218)
- `canSelectPrevious` (page 209)

Declared In
`NSArrayController.h`

canSelectPrevious

Returns `YES` if the previous object, relative to the current selection, in the receiver's content array can be selected.

```
- (BOOL)canSelectPrevious
```

Discussion
The result of this method can be used by a binding to enable user interface items.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `canSelectNext` (page 209)
- `selectPrevious:` (page 218)

Declared In
`NSArrayController.h`

clearsFilterPredicateOnInsertion

Returns a Boolean value that indicates whether the receiver automatically clears an existing filter predicate when new items are inserted or added to the content.

```
- (BOOL)clearsFilterPredicateOnInsertion
```

Return Value
`YES` if the receiver automatically clears an existing filter predicate when new items are inserted or added to the content, otherwise `NO`.

Discussion
The default is `YES`.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setClearsFilterPredicateOnInsertion:` (page 221)

Declared In
`NSArrayController.h`

didChangeArrangementCriteria

Invoked when any criteria for arranging objects change.

```
- (void)didChangeArrangementCriteria
```

Discussion
This method is invoked by the controller itself when any criteria for arranging objects change (sort descriptors or filter predicates) to reset the key paths for automatic rearranging.

Special Considerations

If you implement a subclass of `NSArrayController` and override `rearrangeObjects` (page 213) to use additional arrangement criteria, you should invoke this method if those criteria change.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- rearrangeObjects` (page 213)

Declared In
`NSArrayController.h`

filterPredicate

Returns the predicate used by the receiver to filter the array controller contents.

```
- (NSPredicate *)filterPredicate
```

Return Value
The predicate used by the receiver to filter the array controller contents. Returns `nil` if no filter predicate is set.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setClearsFilterPredicateOnInsertion:` (page 221)
- `setFilterPredicate:` (page 221)

Declared In
`NSArrayController.h`

insert:

Creates a new object and inserts it into the receiver's content array.

```
- (void)insert:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
If an entity name is specified (see `entityName` (page 1749)), this method creates an instance of the of the class specified by the entity, otherwise this method creates an instance of the class specified by `objectClass` (page 1753).

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism (see Error Responders and Error Recovery) can provide feedback as a sheet.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `canInsert` (page 208)

Declared In
`NSArrayController.h`

insertObject:atArrangedObjectIndex:

Inserts *object* into the receiver's arranged objects array at the location specified by *index*, and adds it to the receiver's content collection.

```
- (void)insertObject:(id)object atArrangedObjectIndex:(NSUInteger)index
```

Discussion
Subclasses can override this method to provide customized arranged objects support.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `insertObjects:atArrangedObjectIndexes:` (page 212)
- `addObject:` (page 203)
- `removeObjectAtArrangedObjectIndex:` (page 214)

Related Sample Code
With and Without Bindings

Declared In
NSArrayController.h

insertObjects:atArrangedObjectIndexes:

Inserts *objects* into the receiver's arranged objects array at the locations specified in *indexes*, and adds it to the receiver's content collection.

```
- (void)insertObjects:(NSArray *)objects atArrangedObjectIndexes:(NSIndexSet
    *)indexes
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `insertObject:atArrangedObjectIndex:` (page 211)
- `addObjects:` (page 204)
- `removeObjectsAtArrangedObjectIndexes:` (page 215)

Related Sample Code
DemoMonkey

Declared In
NSArrayController.h

preservesSelection

Returns a Boolean value that indicates whether the receiver will attempt to preserve the current selection when the content changes.

```
- (BOOL)preservesSelection
```

Return Value
`YES` if the receiver attempts to preserve the current selection when the content changes, otherwise `NO`.

Discussion
The default is `YES`.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setClearsFilterPredicateOnInsertion:` (page 221)

Declared In
NSArrayController.h

rearrangeObjects

Triggers filtering of the receiver's content.

```
- (void)rearrangeObjects
```

Discussion
This method invokes `arrangeObjects:` (page 206).

When you detect that filtering criteria change (such as when listening to the text sent by an `NSSearchField` instance), invoke this method on `self`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `arrangeObjects:` (page 206)
- `didChangeArrangementCriteria` (page 210)
- `automaticRearrangementKeyPaths` (page 207)

Related Sample Code
CoreRecipes
Departments and Employees
iSpend

Declared In
`NSArrayController.h`

remove:

Removes the receiver's selected objects from the content collection.

```
- (void)remove:(id)sender
```

Parameters
sender
Typically the object that invoked this method.

Discussion
See removeObject: (page 214) for a discussion of the semantics of removing objects when using Core Data.

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism (see Error Responders and Error Recovery) can provide feedback as a sheet.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `removeObjects:` (page 215)
- `removeObjectAtArrangedObjectIndex:` (page 214)
- `addObject:` (page 203)

Related Sample Code
Aperture Edit Plugin - Borders & Titles
CoreRecipes

Declared In
NSArrayController.h

removeObject:

Removes *object* from the receiver's content collection.

```
- (void)removeObject:(id)object
```

Special Considerations

If you are using Core Data, the exact semantics of this method differ depending on the settings for the array controller. If the receiver's content is fetched automatically, removed objects are marked for deletion by the managed object context (and hence removal from the object graph). If, however, the receiver's `contentSet` is bound to a relationship, `removeObject:` by default only removes the object from the relationship (not from the object graph). You can, though, set the "Deletes Object on Remove" option for the `contentSet` binding, in which case objects are marked for deletion as well as being removed from the relationship.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `removeObjects:` (page 215)
- `removeObjectAtArrangedObjectIndex:` (page 214)
- `addObject:` (page 203)

Related Sample Code
CameraBrowser
ScannerBrowser

Declared In
NSArrayController.h

removeObjectAtArrangedObjectIndex:

Removes the object at the specified *index* in the receiver's arranged objects from the receiver's content array.

```
- (void)removeObjectAtArrangedObjectIndex:(NSUInteger)index
```

Discussion
See removeObject: (page 214) for a discussion of the semantics of removing objects when using Core Data.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `removeObjectsAtArrangedObjectIndexes:` (page 215)
- `insertObject:atArrangedObjectIndex:` (page 211)

- removeObject: (page 214)

Declared In
NSArrayController.h

removeObjects:

Removes *objects* from the receiver's content collection.

- (void)removeObjects:(NSArray *)*objects*

Special Considerations

See removeObject: (page 214) for a discussion of the semantics of removing objects when using Core Data.

Availability
Available in Mac OS X v10.3 and later.

See Also
- removeObject: (page 214)
- removeObjectsAtArrangedObjectIndexes: (page 215)
- addObjects: (page 204)

Related Sample Code
QTCompressionOptionsWindow

Declared In
NSArrayController.h

removeObjectsAtArrangedObjectIndexes:

Removes the objects at the specified *indexes* in the receiver's arranged objects from the content array.

- (void)removeObjectsAtArrangedObjectIndexes:(NSIndexSet *)*indexes*

Discussion
See removeObject: (page 214) for a discussion of the semantics of removing objects when using Core Data.

Availability
Available in Mac OS X v10.3 and later.

See Also
- removeObjectAtArrangedObjectIndex: (page 214)
- insertObjects:atArrangedObjectIndexes: (page 212)
- removeObjects: (page 215)

Declared In
NSArrayController.h

removeSelectedObjects:

Removes *objects* from the receiver's current selection, returning YES if the selection was changed.

```
- (BOOL)removeSelectedObjects:(NSArray *)objects
```

Discussion
Attempting to change the selection may cause a `commitEditing` (page 850) message which fails, thus denying the selection change.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- addSelectedObjects:` (page 204)

Declared In
`NSArrayController.h`

removeSelectionIndexes:

Removes the object as the specified *indexes* from the receiver's current selection, returning `YES` if the selection was changed.

```
- (BOOL)removeSelectionIndexes:(NSIndexSet *)indexes
```

Discussion
Attempting to change the selection may cause a `commitEditing` (page 850) message which fails, thus denying the selection change.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- addSelectionIndexes:` (page 205)

Declared In
`NSArrayController.h`

selectedObjects

Returns an array containing the receiver's selected objects.

```
- (NSArray *)selectedObjects
```

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setSelectedObjects:` (page 222)

Related Sample Code
CalendarItems
CoreRecipes

QTCompressionOptionsWindow
ScannerBrowser

Declared In
NSArrayController.h

selectionIndex

Returns the index of the first object in the receiver's selection.

```
- (NSUInteger)selectionIndex
```

Return Value
The index of the first object in the receiver's selection, or `NSNotFound` if there is no selection.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setSelectionIndex:` (page 222)
- `selectionIndexes` (page 217)

Related Sample Code
ClipboardViewer
CoreRecipes
DemoMonkey
SBSystemPrefs
ScriptingBridgeFinder

Declared In
NSArrayController.h

selectionIndexes

Returns an index set containing the indexes of the receiver's currently selected objects in the content array.

```
- (NSIndexSet *)selectionIndexes
```

Return Value
An index set containing the indexes of the receiver's currently selected objects in the content array.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setSelectionIndexes:` (page 223)

- selectionIndex (page 217)

Declared In
NSArrayController.h

selectNext:

Selects the next object, relative to the current selection, in the receiver's arranged content.

```
- (void)selectNext:(id)sender
```

Discussion
The *sender* is typically the object that invoked this method.

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism (see Error Responders and Error Recovery) can provide feedback as a sheet.

Availability
Available in Mac OS X v10.3 and later.

See Also
- selectPrevious: (page 218)
- canSelectNext (page 209)

Declared In
NSArrayController.h

selectPrevious:

Selects the previous object, relative to the current selection, in the receiver's arranged content.

```
- (void)selectPrevious:(id)sender
```

Discussion
The *sender* is typically the object that invoked this method.

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism (see Error Responders and Error Recovery) can provide feedback as a sheet.

Availability
Available in Mac OS X v10.3 and later.

See Also
- selectNext: (page 218)
- canSelectPrevious (page 209)

Declared In
NSArrayController.h

selectsInsertedObjects

Returns whether the receiver automatically selects inserted objects.

```
- (BOOL)selectsInsertedObjects
```

Return Value
`YES` if the receiver automatically selects inserted objects, otherwise `NO`.

Discussion
The default is `YES`.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setSelectsInsertedObjects:` (page 224)

Declared In
`NSArrayController.h`

setAlwaysUsesMultipleValuesMarker:

Sets whether the receiver always returns the multiple values marker when multiple objects are selected.

```
- (void)setAlwaysUsesMultipleValuesMarker:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver always returns the multiple values marker when multiple objects are selected, even if they have the same value.

Discussion
Setting *flag* to `YES` can increase performance if your application doesn't allow editing multiple values. The default is `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- alwaysUsesMultipleValuesMarker` (page 205)

Declared In
`NSArrayController.h`

setAutomaticallyPreparesContent:

Sets whether the receiver automatically creates and inserts new content objects automatically.

```
- (void)setAutomaticallyPreparesContent:(BOOL)flag
```

Parameters

flag

If YES, the receiver automatically prepares its content.

Discussion

If *flag* is YES and the receiver is not using a managed object context, prepareContent (page 1753) is used to create the content object.

If *flag* is YES and a managed object context is set, the initial content is fetched from the managed object context using the current fetch predicate. The controller also registers as an observer of its managed object context. It then tracks insertions and deletions of its entity using the context's notifications, and updates its content array as appropriate.

Setting *flag* to YES is the same as checking the "Automatically Prepares Content" option in the Interface Builder controller inspector.

See Also

- automaticallyPreparesContent (page 206)
- prepareContent (page 1753)

setAutomaticallyRearrangesObjects:

Sets whether or not the receiver automatically rearranges its content to correspond to the current sort descriptors and filter predicates.

```
- (void)setAutomaticallyRearrangesObjects:(BOOL)flag
```

Parameters

flag

A Boolean value that indicates whether the receiver automatically rearranges its content (YES) or not (NO).

Discussion

The default is NO.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSArrayController.h

setAvoidsEmptySelection:

Sets whether the receiver attempts to avoid an empty selection.

```
- (void)setAvoidsEmptySelection:(BOOL)flag
```

Parameters

flag

If YES, the receiver maintains a selection unless there are no objects in the content array.

Discussion

The default is YES.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `avoidsEmptySelection` (page 208)

Declared In
`NSArrayController.h`

setClearsFilterPredicateOnInsertion:

Sets whether the receiver automatically clears an existing filter predicate when a new object is inserted or added to the content array.

```
- (void)setClearsFilterPredicateOnInsertion:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver automatically clears an existing filter predicate when a new object is inserted or added to the content array.

Discussion
The default is `YES`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `clearsFilterPredicateOnInsertion` (page 209)

Declared In
`NSArrayController.h`

setFilterPredicate:

Sets the predicate used to filter the contents of the receiver.

```
- (void)setFilterPredicate:(NSPredicate *)filterPredicate
```

Parameters

filterPredicate

The predicate used to filter the contents of the receiver.

Discussion
If *filterPredicate* is `nil`, any existing filter predicate is removed.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `filterPredicate` (page 210)

Declared In
`NSArrayController.h`

setPreservesSelection:

Sets whether the receiver attempts to preserve selection when the content changes.

```
- (void)setPreservesSelection:(BOOL)flag
```

Parameters

flag

If YES, the receiver attempts to preserve selection when the content changes.

Discussion
The default is YES.

Availability
Available in Mac OS X v10.3 and later.

See Also
- preservesSelection (page 212)

Declared In
NSArrayController.h

setSelectedObjects:

Sets *objects* as the receiver's current selection, returning YES if the selection was changed.

```
- (BOOL)setSelectedObjects:(NSArray *)objects
```

Discussion
Attempting to change the selection may cause a commitEditing (page 850) message which fails, thus denying the selection change.

Availability
Available in Mac OS X v10.3 and later.

See Also
- selectedObjects (page 216)
- addSelectedObjects: (page 204)

Related Sample Code
CoreRecipes

Declared In
NSArrayController.h

setSelectionIndex:

Sets the receiver's selection to the given index, and returns a Boolean value that indicates whether the selection was changed.

```
- (BOOL)setSelectionIndex:(NSUInteger)index
```

Parameters

index

The index for the selection.

Return Value

`YES` if the selection was changed, otherwise `NO`.

Discussion

Attempting to change the selection may cause a `commitEditing` (page 850) message which fails, thus denying the selection change.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `selectionIndex` (page 217)
- `setSelectionIndexes:` (page 223)

Related Sample Code

CoreRecipes

DemoMonkey

GridCalendar

QTCompressionOptionsWindow

Declared In

`NSArrayController.h`

setSelectionIndexes:

Sets the receiver's selection indexes and returns a Boolean value that indicates whether the selection changed.

```
- (BOOL)setSelectionIndexes:(NSIndexSet *)indexes
```

Parameters

indexes

The set of selection indexes for the receiver.

Return Value

`YES` if the selection was changed, otherwise `NO`.

Discussion

Attempting to change the selection may cause a `commitEditing` (page 850) message which fails, thus denying the selection change.

To select all the receiver's objects, indexes should be an index set with indexes `[0...count -1]`. To deselect all indexes, pass an empty index set.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `selectionIndexes` (page 217)
- `setSelectionIndex:` (page 222)

Related Sample Code
DemoMonkey

Declared In
`NSArrayController.h`

setSelectsInsertedObjects:

Sets whether the receiver will automatically select objects as they are inserted.

```
- (void)setSelectsInsertedObjects:(BOOL)flag
```

Parameters

flag
If `YES` then items will be selected upon insertion.

Discussion
The default is `YES`.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- selectsInsertedObjects` (page 219)

Declared In
`NSArrayController.h`

setSortDescriptors:

Sets the sort descriptors for the receiver.

```
- (void)setSortDescriptors:(NSArray *)sortDescriptors
```

Parameters

sortDescriptors
An array of `NSSortDescriptor` objects, used by the receiver to arrange its content.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- sortDescriptors` (page 224)
`- arrangeObjects:` (page 206)

Declared In
`NSArrayController.h`

sortDescriptors

Returns the receiver's array of sort descriptors.

```
- (NSArray *)sortDescriptors
```

Return Value
The array of `NSSortDescriptor` objects used by the receiver to arrange its content.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setSortDescriptors:` (page 224)
- `arrangeObjects:` (page 206)

Related Sample Code
iSpend

Declared In
`NSArrayController.h`

NSATSTypesetter Class Reference

Inherits from	NSTypesetter : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSATSTypesetter.h
Availability	Available in Mac OS X v10.3 and later.
Companion guides	Text System Overview Text Layout Programming Guide

Overview

`NSATSTypesetter` is a concrete subclass of `NSTypesetter` that places glyphs during the text layout process. The typesetter creates line fragment rectangles, positions glyphs within the line fragments, determines line breaks by word wrapping and hyphenation, and handles tab positioning.

`NSATSTypesetter` encapsulates the advanced typesetting capabilities of Core Text. `NSATSTypesetter` provides enhanced line and character spacing accuracy and supports more languages, including bidirectional languages, than the original, built-in typesetter class `NSSimpleHorizontalTypesetter` (which is deprecated in Mac OS X version 10.4 and later).

Subclassing Notes

`NSATSTypesetter` introduced a set of interfaces in Mac OS X version 10.3 that facilitated subclassing and made it possible to substitute a custom layout engine into the Cocoa text system. In Mac OS X version 10.4, those interfaces moved to `NSTypesetter`, which you can subclass to the same effect. See the `NSTypesetter` reference documentation for relevant subclassing notes.

Tasks

Getting a Typesetter

`+ sharedTypesetter` (page 231)
Returns a shared instance of `NSATSTypesetter`.

Managing the Layout Manager

- `layoutManager` (page 237)
  Returns the layout manager for the text being typeset.
- `setUsesFontLeading:` (page 245)
  Sets a Boolean value controlling whether the typesetter uses the leading (or line gap) value specified in the font metric information.
- `usesFontLeading` (page 247)
  Returns a Boolean value indicating whether the typesetter uses the leading (or line gap) value specified in the font metric information of the current font.
- `setTypesetterBehavior:` (page 244)
  Sets the default typesetter behavior, which affects glyph spacing and line height.
- `typesetterBehavior` (page 247)
  Returns the current typesetter behavior value.
- `setHyphenationFactor:` (page 242)
  Sets the threshold controlling when hyphenation is attempted
- `hyphenationFactor` (page 235)
  Returns the current hyphenation factor.

Managing the Text Container

- `currentTextContainer` (page 233)
  Returns the text container for the text being typeset.
- `setLineFragmentPadding:` (page 242)
  Sets the amount (in points) by which text is inset within line fragment rectangles
- `lineFragmentPadding` (page 237)
  Returns the current line fragment padding amount; that is, the portion on each end of the line fragment rectangle left blank.

Mapping Screen and Printer Fonts

- `substituteFontForFont:` (page 246)
  Returns a screen font suitable for use in place of the specified original font, or simply returns the original font if a screen font can't be used or isn't available.

Managing Text Tabs

- `textTabForGlyphLocation:writingDirection:maxLocation:` (page 247)
  Returns the text tab next closest to a given glyph location, indexing in the specified direction but not beyond a given glyph location.

Bidirectional Text Processing

- `setBidiProcessingEnabled:` (page 241)
  Sets a Boolean value controlling whether the typesetter performs bidirectional text processing.
- `bidiProcessingEnabled` (page 232)
  Returns a Boolean value indicating the bidirectional text processing setting currently in effect.

Accessing Paragraph Typesetting Information

- `setAttributedString:` (page 240)
  Sets the text backing store on which this typesetter operates.
- `attributedString` (page 231)
  Returns the text backing store, usually an instance of `NSTextStorage`.
- `setParagraphGlyphRange:separatorGlyphRange:` (page 244)
  Sets the current glyph range being processed and the paragraph separator glyph range (the range of the paragraph separator character or characters).
- `paragraphGlyphRange` (page 238)
  Returns the glyph range currently being processed.
- `paragraphSeparatorGlyphRange` (page 239)
  Returns the current paragraph separator range, which is the full range that contains the current glyph range and that extends from one paragraph separator character to the next.

Paragraph Layout

- `layoutParagraphAtPoint:` (page 237)
  Lays out glyphs in the current glyph range until the next paragraph separator is reached.

Line and Paragraph Spacing

- `lineSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:` (page 238)
  Returns the line spacing in effect following the specified glyph.
- `paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:` (page 239)
  Returns the paragraph spacing, the number of points of space added following a paragraph, which is in effect after the specified glyph.
- `paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:` (page 240)
  Returns the number of points of space added before a paragraph, which is in effect before the specified glyph.

Glyph Caching

- `setHardInvalidation:forGlyphRange:` (page 242)
  Sets a Boolean value controlling whether to force the layout manager to invalidate the portion of the glyph cache in the given glyph range when invalidating layout.

Laying out Glyphs

`- boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:` (page 232)

Returns the bounding rectangle for the given control glyph, at the given glyph position and character index, in the given text container.

`- getLineFragmentRect:usedRect:forParagraphSeparatorGlyphRange:atProposedOrigin:` (page 234)

Calculates the line fragment rectangle and line fragment used rectangle for blank lines.

`- hyphenCharacterForGlyphAtIndex:` (page 236)

Returns the hyphen character to be inserted after the given glyph when hyphenation is enabled in the layout manager.

`- hyphenationFactorForGlyphAtIndex:` (page 235)

Returns the hyphenation factor in effect at the given glyph index.

`- shouldBreakLineByHyphenatingBeforeCharacterAtIndex:` (page 245)

The typesetter calls this method, if implemented by a subclass, before breaking a line by hyphenating before the character at the given character index, enabling the subclass to control line breaking.

`- shouldBreakLineByWordBeforeCharacterAtIndex:` (page 246)

The typesetter calls this method, if implemented by a subclass, before breaking a line by word wrapping before the character at the given character index, enabling the subclass to control line breaking.

`- willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 248)

Called by the typesetter just prior to calling `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 243) which stores the actual line fragment rectangle location in the layout manager.

Interfacing with Glyph Storage

`- characterRangeForGlyphRange:actualGlyphRange:` (page 232)

Returns the range for the characters in the receiver's text store that are mapped to the glyphs in the given glyph range.

`- deleteGlyphsInRange:` (page 233)

Deletes the glyphs in the given glyph range from the glyph cache maintained by the layout manager.

`- getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:` (page 233)

Extracts the information needed to lay out the glyphs in the given glyph buffer from the given glyph range.

`- glyphRangeForCharacterRange:actualCharacterRange:` (page 234)

Returns the range for the glyphs mapped to the characters of the text store in the given character range.

`- insertGlyph:atGlyphIndex:characterIndex:` (page 236)

Enables the typesetter to insert a new glyph into the stream.

`- setAttachmentSize:forGlyphRange:` (page 240)

Sets the size the glyphs in the given glyph range (assumed to be attachments) will be asked to draw themselves.

`- setBidiLevels:forGlyphRange:` (page 241)

Sets the direction of the glyphs in the given glyph range for bidirectional text to the given levels.

- `setDrawsOutsideLineFragment:forGlyphRange:` (page 241)
  Sets a Boolean value controlling whether the glyphs in the given glyph range exceed the bounds of the line fragment in which they are laid out.
- `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 243)
  Sets the line fragment rectangle where the glyphs in in the given glyph range are laid out to the given line fragment rectangle.
- `setLocation:withAdvancements:forStartOfGlyphRange:` (page 243)
  Sets the location where the glyphs in the given glyph range are laid out to the specified location.
- `setNotShownAttribute:forGlyphRange:` (page 244)
  Sets a Boolean value controlling whether the glyphs in the given glyph rangeare not shown.
- `substituteGlyphsInRange:withGlyphs:` (page 246)
  Replaces the glyphs in the given glyph range with the given glyphs.
- `lineFragmentRectForProposedRect:remainingRect:` (page 238) Deprecated in Mac OS X v10.4
  This method has been deprecated. Use the NSTypesetter method `getLineFragmentRect:usedRect:remainingRect:forStartingGlyphAtIndex:proposedRect:lineSpacing:paragraphSpacingBefore:paragraphSpacingAfter:` (page 3090) instead.

Class Methods

sharedTypesetter

Returns a shared instance of `NSATSTypesetter`.

```
+ (id)sharedTypesetter
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSATSTypesetter.h`

Instance Methods

attributedString

Returns the text backing store, usually an instance of `NSTextStorage`.

```
- (NSAttributedString *)attributedString
```

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- `setAttributedString:` (page 240)

Declared In
NSATSTypesetter.h

bidiProcessingEnabled

Returns a Boolean value indicating the bidirectional text processing setting currently in effect.

```
- (BOOL)bidiProcessingEnabled
```

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- setBidiProcessingEnabled: (page 241)

Declared In
NSATSTypesetter.h

boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:

Returns the bounding rectangle for the given control glyph, at the given glyph position and character index, in the given text container.

```
- (NSRect)boundingBoxForControlGlyphAtIndex:(NSUInteger)glyphIndex
    forTextContainer:(NSTextContainer *)textContainer
    proposedLineFragment:(NSRect)proposedRect glyphPosition:(NSPoint)glyphPosition
     characterIndex:(NSUInteger)charIndex
```

Discussion
Returns the bounding rectangle for the control glyph at *glyphIndex*, at the given *glyphPosition* and character index *charIndex*, in *textContainer*. The proposed line fragment rectangle is specified by *proposedRect*.

The typesetter calls this method when it encounters an NSControlGlyph. The default behavior is to return zero width for control glyphs. A subclass can override this method to do something different, such as implement a way to display control characters.

NSGlyphGenerator can choose whether or not to map control characters to NSControlGlyph. Tab characters, for example, do not use this facility.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

characterRangeForGlyphRange:actualGlyphRange:

Returns the range for the characters in the receiver's text store that are mapped to the glyphs in the given glyph range.

```
- (NSRange)characterRangeForGlyphRange:(NSRange)glyphRange
    actualGlyphRange:(NSRangePointer)actualGlyphRange
```

Discussion
If *actualGlyphRange* is non-NULL, expands the requested range as needed so that it identifies all glyphs mapped to those characters and returns the new range by reference in *actualGlyphRange*.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- glyphRangeForCharacterRange:actualCharacterRange: (page 234)

Declared In
NSATSTypesetter.h

currentTextContainer

Returns the text container for the text being typeset.

```
- (NSTextContainer *)currentTextContainer
```

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

deleteGlyphsInRange:

Deletes the glyphs in the given glyph range from the glyph cache maintained by the layout manager.

```
- (void)deleteGlyphsInRange:(NSRange)glyphRange
```

Discussion
A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
insertGlyph:atGlyphIndex:characterIndex: (page 236)

Declared In
NSATSTypesetter.h

getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:

Extracts the information needed to lay out the glyphs in the given glyph buffer from the given glyph range.

```
- (NSUInteger)getGlyphsInRange:(NSRange)glyphsRange glyphs:(NSGlyph *)glyphBuffer
    characterIndexes:(NSUInteger *)charIndexBuffer
  glyphInscriptions:(NSGlyphInscription *)inscribeBuffer elasticBits:(BOOL
  *)elasticBuffer
```

Discussion
The *charIndexBuffer* contains the original characters for the glyphs. Note that a glyph at index 1 is not necessarily mapped to the character at index 1, since a glyph may be for a ligature or accent.

The *inscribeBuffer* contains the inscription attributes for each glyph, which are used to layout characters that are combined together. The possible values are described in the "Constants" (page 1525) section of the NSLayoutManager reference.

The *elasticBuffer* contains a Boolean value indicating whether a glyph is elastic for each glyph. An elastic glyph can be made longer at the end of a line or when needed for justification.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

getLineFragmentRect:usedRect:forParagraphSeparatorGlyphRange:atProposedOrigin:

Calculates the line fragment rectangle and line fragment used rectangle for blank lines.

```
- (void)getLineFragmentRect:(NSRect *)lineFragmentRect usedRect:(NSRect
  *)lineFragmentUsedRect
  forParagraphSeparatorGlyphRange:(NSRange)paragraphSeparatorGlyphRange
  atProposedOrigin:(NSPoint)lineOrigin
```

Discussion
The method returns the calculated line fragment rectangle in *lineFragmentRect*, and it returns the used rectangle (the portion of the line fragment rectangle that actually contains marks) in *lineFragmentUsedRect*. The *paragraphSeparatorGlyphRange* is the range of glyphs under consideration, and *lineOrigin* is the origin point of the line fragment rectangle. A *paragraphSeparatorGlyphRange* with length 0 indicates an extra line fragment (which occurs if the last character in the paragraph is a line separator.)

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

glyphRangeForCharacterRange:actualCharacterRange:

Returns the range for the glyphs mapped to the characters of the text store in the given character range.

```
- (NSRange)glyphRangeForCharacterRange:(NSRange)charRange
    actualCharacterRange:(NSRangePointer)actualCharRange
```

Discussion
If *actualCharRange* is non-NULL, expands the requested range as needed so that it identifies all characters mapped to those glyphs and returns the new range by reference in *actualCharRange*.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- characterRangeForGlyphRange:actualGlyphRange: (page 232)

Declared In
NSATSTypesetter.h

hyphenationFactor

Returns the current hyphenation factor.

```
- (float)hyphenationFactor
```

Discussion
The hyphenation factor is a value ranging from 0.0 to 1.0 that controls when hyphenation is attempted. By default, the value is 0.0, meaning hyphenation is off. A factor of 1.0 causes hyphenation to be attempted always.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- setHyphenationFactor: (page 242)

Declared In
NSATSTypesetter.h

hyphenationFactorForGlyphAtIndex:

Returns the hyphenation factor in effect at the given glyph index.

```
- (float)hyphenationFactorForGlyphAtIndex:(NSUInteger)glyphIndex
```

Discussion
The hyphenation factor is a value ranging from 0.0 to 1.0 that controls when hyphenation is attempted. By default, the value is 0.0, meaning hyphenation is off. A factor of 1.0 causes hyphenation to be attempted always.

The typesetter calls this method with a proposed hyphenation point for a line break to find the hyphenation factor in effect at that time. A subclass can override this method to customize the text layout process.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- hyphenCharacterForGlyphAtIndex: (page 236)

Declared In
NSATSTypesetter.h

hyphenCharacterForGlyphAtIndex:

Returns the hyphen character to be inserted after the given glyph when hyphenation is enabled in the layout manager.

```
- (UTF32Char)hyphenCharacterForGlyphAtIndex:(NSUInteger)glyphIndex
```

Discussion
The typesetter calls this method before hyphenating. A subclass can override this method to return a different hyphen glyph.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- hyphenationFactorForGlyphAtIndex: (page 235)

Declared In
NSATSTypesetter.h

insertGlyph:atGlyphIndex:characterIndex:

Enables the typesetter to insert a new glyph into the stream. (Available in Mac OS X v10.3 through Mac OS X v10.3.)

```
- (void)insertGlyph:(NSGlyph)glyph atGlyphIndex:(NSUInteger)glyphIndex
    characterIndex:(NSUInteger)charIndex
```

Discussion
Inserts *glyph* into the glyph cache at *glyphIndex* and maps it to the character at *charIndex*. If the glyph is mapped to several characters, *charIndex* should indicate the first character to which it's mapped.

The standard typesetter uses this method for inserting hyphenation glyphs. Because this method keeps the glyph caches synchronized, subclasses should always use this method to insert glyphs instead of calling layoutManager (page 237) directly.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3 through Mac OS X v10.3.

Declared In
NSATSTypesetter.h

layoutManager

Returns the layout manager for the text being typeset.

```
- (NSLayoutManager *)layoutManager
```

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

layoutParagraphAtPoint:

Lays out glyphs in the current glyph range until the next paragraph separator is reached.

```
- (NSUInteger)layoutParagraphAtPoint:(NSPoint *)lineFragmentOrigin
```

Discussion
The *lineFragmentOrigin* specifies the upper-left corner of line fragment rectangle. On return, *lineFragmentOrigin* contains the next origin. This method returns the next glyph index. Usually it's the index right after the paragraph separator, but it can be inside the paragraph range if, for example, the end of the text container is reached before the paragraph separator.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

lineFragmentPadding

Returns the current line fragment padding amount; that is, the portion on each end of the line fragment rectangle left blank.

```
- (CGFloat)lineFragmentPadding
```

Discussion
Text is inset within the line fragment rectangle by this amount.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- setLineFragmentPadding: (page 242)

Declared In
NSATSTypesetter.h

lineFragmentRectForProposedRect:remainingRect:

This method has been deprecated. Use the NSTypesetter method `getLineFragmentRect:usedRect:remainingRect:forStartingGlyphAtIndex:proposedRect:lineSpacing:paragraphSpacingBefore:paragraphSpacingAfter:` (page 3090) instead. (Deprecated in Mac OS X v10.4.)

```
- (NSRect)lineFragmentRectForProposedRect:(NSRect)proposedRect
    remainingRect:(NSRectPointer)remainingRect
```

Discussion
Returns the largest rectangle available for the proposed rectangle *proposedRect*. It also returns a rectangle in *remainingRect* containing any remaining space, such as that left on the other side of a hole or gap in the text container.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3 and later.
Deprecated in Mac OS X v10.4.

Declared In
`NSATSTypesetter.h`

lineSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:

Returns the line spacing in effect following the specified glyph.

```
- (CGFloat)lineSpacingAfterGlyphAtIndex:(NSUInteger)glyphIndex
    withProposedLineFragmentRect:(NSRect)rect
```

Discussion
The NSATSTypesetter calls this method to determine the number of points of space to include below the descenders in the used rectangle for the proposed line fragment rectangle *rect*.

Line spacing, also called leading, is an attribute of NSParagraphStyle, which you can set on an NSMutableParagraphStyle object. A font typically includes a default minimum line spacing metric used if none is set in the paragraph style.

If the typesetter behavior specified in the NSLayoutManager is `NSTypesetterOriginalBehavior`, the text system uses the original, private typesetter NSSimpleHorizontalTypesetter, which adds the line spacing above the ascender. Similarly, NSATSTypesetter adds the line spacing above the ascender if the value is negative.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
`NSATSTypesetter.h`

paragraphGlyphRange

Returns the glyph range currently being processed.

```
- (NSRange)paragraphGlyphRange
```

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- `setParagraphGlyphRange:separatorGlyphRange:` (page 244)
- `paragraphSeparatorGlyphRange` (page 239)

Declared In
`NSATSTypesetter.h`

paragraphSeparatorGlyphRange

Returns the current paragraph separator range, which is the full range that contains the current glyph range and that extends from one paragraph separator character to the next.

```
- (NSRange)paragraphSeparatorGlyphRange
```

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- `setParagraphGlyphRange:separatorGlyphRange:` (page 244)
- `paragraphGlyphRange` (page 238)

Declared In
`NSATSTypesetter.h`

paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:

Returns the paragraph spacing, the number of points of space added following a paragraph, which is in effect after the specified glyph.

```
- (CGFloat)paragraphSpacingAfterGlyphAtIndex:(NSUInteger)glyphIndex
    withProposedLineFragmentRect:(NSRect)rect
```

Discussion
The *rect* argument specifies the line fragment rectangle of the last line in the paragraph.

The typesetter adds the number of points specified in the return value to the bottom of the line fragment rectangle specified by *rect* (but not to the used line fragment rectangle for that line). Paragraph spacing added after a paragraph correlates to the value returned by the `paragraphSpacing` method of NSParagraphStyle, which you can set using the `setParagraphSpacing:` method of NSMutableParagraphStyle.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- `paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:` (page 240)

Declared In
`NSATSTypesetter.h`

paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:

Returns the number of points of space added before a paragraph, which is in effect before the specified glyph.

```
- (CGFloat)paragraphSpacingBeforeGlyphAtIndex:(NSUInteger)glyphIndex
    withProposedLineFragmentRect:(NSRect)rect
```

Discussion
The *rect* argument specifies the line fragment rectangle of the first line in the paragraph.

The typesetter adds the number of points specified in the return value to the top of the line fragment rectangle specified by *rect* (but not to the used line fragment rectangle for that line). Paragraph spacing added before a paragraph correlates to the value returned by the `paragraphSpacingBefore` method of NSParagraphStyle, which you can set using the `setParagraphSpacingBefore:` method of NSMutableParagraphStyle.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
`- paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:` (page 239)

Declared In
`NSATSTypesetter.h`

setAttachmentSize:forGlyphRange:

Sets the size the glyphs in the given glyph range (assumed to be attachments) will be asked to draw themselves.

```
- (void)setAttachmentSize:(NSSize)attachmentSize forGlyphRange:(NSRange)glyphRange
```

Discussion
A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
`NSATSTypesetter.h`

setAttributedString:

Sets the text backing store on which this typesetter operates.

```
- (void)setAttributedString:(NSAttributedString *)attrString
```

Discussion
The string object is not retained.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- attributedString (page 231)

Declared In
NSATSTypesetter.h

setBidiLevels:forGlyphRange:

Sets the direction of the glyphs in the given glyph range for bidirectional text to the given levels.

```
- (void)setBidiLevels:(const uint8_t *)levels forGlyphRange:(NSRange)glyphRange
```

Discussion
The value of *levels* can range from 0 to 61 as defined by Unicode Standard Annex #9.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

setBidiProcessingEnabled:

Sets a Boolean value controlling whether the typesetter performs bidirectional text processing.

```
- (void)setBidiProcessingEnabled:(BOOL)flag
```

Discussion
You can use this method to disable the bidirectional layout stage if you know the paragraph does not need this stage; that is, if the characters in the backing store are in display order.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- bidiProcessingEnabled (page 232)

Declared In
NSATSTypesetter.h

setDrawsOutsideLineFragment:forGlyphRange:

Sets a Boolean value controlling whether the glyphs in the given glyph range exceed the bounds of the line fragment in which they are laid out.

```
- (void)setDrawsOutsideLineFragment:(BOOL)flag forGlyphRange:(NSRange)glyphRange
```

Discussion
This can happen when text is set at a fixed line height. For example, if the user specifies a fixed line height of 12 points and sets the font size to 24 points, the glyphs will exceed their layout rectangles.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

setHardInvalidation:forGlyphRange:

Sets a Boolean value controlling whether to force the layout manager to invalidate the portion of the glyph cache in the given glyph range when invalidating layout.

```
- (void)setHardInvalidation:(BOOL)flag forGlyphRange:(NSRange)glyphRange
```

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

setHyphenationFactor:

Sets the threshold controlling when hyphenation is attempted

```
- (void)setHyphenationFactor:(float)factor
```

Discussion
The *factor* argument is in the range of 0.0 to 1.0. By default, the value is 0.0, meaning hyphenation is off. A *factor* of 1.0 causes hyphenation to be attempted always.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- hyphenationFactor (page 235)

Declared In
NSATSTypesetter.h

setLineFragmentPadding:

Sets the amount (in points) by which text is inset within line fragment rectangles

```
- (void)setLineFragmentPadding:(CGFloat)padding
```

Discussion
Note that line fragment padding isn't a suitable means for expressing margins; you should set the NSTextView object's position and size for document margins or the paragraph margin attributes for text margins.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- `lineFragmentPadding` (page 237)

Declared In
`NSATSTypesetter.h`

setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:

Sets the line fragment rectangle where the glyphs in in the given glyph range are laid out to the given line fragment rectangle.

```
- (void)setLineFragmentRect:(NSRect)fragmentRect forGlyphRange:(NSRange)glyphRange
    usedRect:(NSRect)usedRect baselineOffset:(CGFloat)baselineOffset
```

Discussion
The exact positions of the glyphs must be set after the line fragment rectangle with `setLocation:forStartOfGlyphRange:`.

The *usedRect* argument indicates the portion of *fragmentRect*, in the NSTextContainer object's coordinate system, that actually contains glyphs or other marks that are drawn (including the text container's line fragment padding). The *usedRect* must be equal to or contained within *fragmentRect*. The *baselineOffset* argument is the vertical distance in pixels from the line fragment origin to the baseline on which the glyphs align.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
`NSATSTypesetter.h`

setLocation:withAdvancements:forStartOfGlyphRange:

Sets the location where the glyphs in the given glyph range are laid out to the specified location.

```
- (void)setLocation:(NSPoint)location withAdvancements:(const CGFloat *)advancements
    forStartOfGlyphRange:(NSRange)glyphRange
```

Discussion
The x-coordinate of *location* is expressed relative to the line fragment rectangle origin, and the y-coordinate is expressed relative to the baseline previously specified by `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 243). The *glyphRange* defines a series of glyphs that can be displayed with a single PostScript `show` operation (a nominal range). Setting the location for a series of glyphs implies that the glyphs preceding it can't be included in a single `show` operation. The *advancements* argument is the nominal glyph advance width specified in the font metric information.

Before setting the location for a glyph range, you must specify line fragment rectangle with `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:`.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

setNotShownAttribute:forGlyphRange:

Sets a Boolean value controlling whether the glyphs in the given glyph rangeare not shown.

```
- (void)setNotShownAttribute:(BOOL)flag forGlyphRange:(NSRange)glyphRange
```

Discussion
For example, a tab or newline character doesn't leave any marks; it just indicates where following glyphs are laid out.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

setParagraphGlyphRange:separatorGlyphRange:

Sets the current glyph range being processed and the paragraph separator glyph range (the range of the paragraph separator character or characters).

```
- (void)setParagraphGlyphRange:(NSRange)paragraphRange
    separatorGlyphRange:(NSRange)paragraphSeparatorRange
```

Parameters

paragraphRange
The glyph range that becomes current.

paragraphSeparatorRange
The paragraph separator glyph range that becomes current.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- paragraphGlyphRange (page 238)
- paragraphSeparatorGlyphRange (page 239)

Declared In
NSATSTypesetter.h

setTypesetterBehavior:

Sets the default typesetter behavior, which affects glyph spacing and line height.

```
- (void)setTypesetterBehavior:(NSTypesetterBehavior)behavior
```

Discussion
The possible values for *behavior* are described in the "Constants" (page 1525) section of the NSLayoutManager reference.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- typesetterBehavior (page 247)

Declared In
NSATSTypesetter.h

setUsesFontLeading:

Sets a Boolean value controlling whether the typesetter uses the leading (or line gap) value specified in the font metric information.

```
- (void)setUsesFontLeading:(BOOL)flag
```

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- usesFontLeading (page 247)

Declared In
NSATSTypesetter.h

shouldBreakLineByHyphenatingBeforeCharacterAtIndex:

The typesetter calls this method, if implemented by a subclass, before breaking a line by hyphenating before the character at the given character index, enabling the subclass to control line breaking.

```
- (BOOL)shouldBreakLineByHyphenatingBeforeCharacterAtIndex:(NSUInteger)charIndex
```

Discussion
A subclass can override this method to customize the text layout process. If the method returns NO, the typesetter continues looking for a break point.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- shouldBreakLineByWordBeforeCharacterAtIndex: (page 246)

Declared In
NSATSTypesetter.h

shouldBreakLineByWordBeforeCharacterAtIndex:

The typesetter calls this method, if implemented by a subclass, before breaking a line by word wrapping before the character at the given character index, enabling the subclass to control line breaking.

```
- (BOOL)shouldBreakLineByWordBeforeCharacterAtIndex:(NSUInteger)charIndex
```

Discussion
A subclass can override this method to customize the text layout process. If the method returns `NO`, the typesetter continues looking for a break point.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- shouldBreakLineByHyphenatingBeforeCharacterAtIndex: (page 245)

Declared In
NSATSTypesetter.h

substituteFontForFont:

Returns a screen font suitable for use in place of the specified original font, or simply returns the original font if a screen font can't be used or isn't available.

```
- (NSFont *)substituteFontForFont:(NSFont *)originalFont
```

Discussion
A screen font can be substituted if the receiver is set to use screen fonts and if no NSTextView associated with the receiver is scaled or rotated.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

substituteGlyphsInRange:withGlyphs:

Replaces the glyphs in the given glyph range with the given glyphs.

```
- (void)substituteGlyphsInRange:(NSRange)glyphRange withGlyphs:(NSGlyph *)glyphs
```

Discussion
This method does not alter the glyph-to-character mapping or invalidate layout information.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSATSTypesetter.h

textTabForGlyphLocation:writingDirection:maxLocation:

Returns the text tab next closest to a given glyph location, indexing in the specified direction but not beyond a given glyph location.

```
- (NSTextTab *)textTabForGlyphLocation:(CGFloat)glyphLocation
    writingDirection:(NSWritingDirection)direction maxLocation:(CGFloat)maxLocation
```

Discussion
The typesetter calls this method whenever it finds a tab character. To determine the width to advance the next glyph, the typesetter examines the NSParagraphStyle tab array and the default tab interval.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In
NSATSTypesetter.h

typesetterBehavior

Returns the current typesetter behavior value.

```
- (NSTypesetterBehavior)typesetterBehavior
```

Discussion
The possible return values are described in the "Constants" (page 1525) section of the `NSLayoutManager` reference.

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- `setTypesetterBehavior:` (page 244)

Declared In
NSATSTypesetter.h

usesFontLeading

Returns a Boolean value indicating whether the typesetter uses the leading (or line gap) value specified in the font metric information of the current font.

```
- (BOOL)usesFontLeading
```

Availability
Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

See Also
- `setUsesFontLeading:` (page 245)

Declared In
NSATSTypesetter.h

willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset:

Called by the typesetter just prior to calling `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 243) which stores the actual line fragment rectangle location in the layout manager.

```
- (void)willSetLineFragmentRect:(NSRect *)lineRect forGlyphRange:(NSRange)glyphRange
    usedRect:(NSRect *)usedRect baselineOffset:(CGFloat *)baselineOffset
```

Discussion

The *lineRect* argument is the rectangle in which the glyphs in *glyphRange* are laid out. The *usedRect* argument indicates the portion of *lineRect*, in the NSTextContainer object's coordinate system, that actually contains glyphs or other marks that are drawn (including the text container's line fragment padding). The *usedRect* must be equal to or contained within *lineRect*. The *baselineOffset* argument is the vertical distance in pixels from the line fragment origin to the baseline on which the glyphs align.

A subclass can override this method to customize the text layout process. For example, it could change the shape of the line fragment rectangle. The subclass is responsible for ensuring that the modified rectangle remains valid (for example, that it lies within the text container).

Availability

Available in Mac OS X v10.3. Moved to NSTypesetter in Mac OS X v10.4 and later.

Declared In

`NSATSTypesetter.h`

NSAttributedString Application Kit Additions Reference

Inherits from	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSAttributedString.h AppKit/NSStringDrawing.h AppKit/NSTextAttachment.h
Companion guide	Attributed String Programming Guide

Overview

The Application Kit extends Foundation's `NSAttributedString` class by adding support for RTF (with or without attachments), graphics attributes (including font and ruler attributes), methods for drawing attributed strings, and methods for calculating significant linguistic units.

Tasks

Creating an NSAttributedString

`+ attributedStringWithAttachment:` (page 252)
Creates an attributed string with an attachment.

`- initWithData:options:documentAttributes:error:` (page 260)
Initializes and returns a new `NSAttributedString` object from the data contained in the given `NSData` object.

`- initWithDocFormat:documentAttributes:` (page 261)
Initializes and returns a new `NSAttributedString` object from Microsoft Word format data contained in the given `NSData` object.

`- initWithHTML:documentAttributes:` (page 261)
Initializes and returns a new `NSAttributedString` object from HTML contained in the given data object.

`- initWithHTML:baseURL:documentAttributes:` (page 261)
Initializes and returns a new `NSAttributedString` object from the HTML contained in the given object and base URL.

`- initWithHTML:options:documentAttributes:` (page 262)
Initializes and returns a new `NSAttributedString` object from HTML contained in the given data object.

- `initWithPath:documentAttributes:` (page 262)
  Initializes a new `NSAttributedString` object from RTF or RTFD data contained in the file at the given path.
- `initWithRTF:documentAttributes:` (page 263)
  Initializes a new `NSAttributedString` object by decoding the stream of RTF commands and data contained in the given data object.
- `initWithRTFD:documentAttributes:` (page 263)
  Initializes a new `NSAttributedString` object by decoding the stream of RTFD commands and data contained in the given data object.
- `initWithRTFDFileWrapper:documentAttributes:` (page 263)
  Initializes a new `NSAttributedString` object from the given `NSFileWrapper` object containing an RTFD document.
- `initWithURL:documentAttributes:` (page 264)
  Initializes a new `NSAttributedString` object from the data at the given URL.
- `initWithURL:options:documentAttributes:error:` (page 264)
  Initializes a new `NSAttributedString` object from the contents of the given URL.

Retrieving Font Attribute Information

- `containsAttachments` (page 256)
  Returns `YES` if the receiver contains any attachment attributes, `NO` otherwise.
- `fontAttributesInRange:` (page 260)
  Returns the font attributes in effect for the character at the given location.
- `rulerAttributesInRange:` (page 269)
  Returns the ruler (paragraph) attributes in effect for the characters within the given range.

Calculating Linguistic Units

- `URLAtIndex:effectiveRange:` (page 270)
  Returns a URL, either from a link attribute or from text at the given location that appears to be a URL string, for use in automatic link detection.
- `doubleClickAtIndex:` (page 257)
  Returns the range of characters that form a word (or other linguistic unit) surrounding the given index, taking language characteristics into account.
- `lineBreakBeforeIndex:withinRange:` (page 265)
  Returns the index of the closest character before the given index, and within the given range, that can be placed on a new line when laying out text.
- `lineBreakByHyphenatingBeforeIndex:withinRange:` (page 265)
  Returns the index of the closest character before the given index, and within the given range, that can be placed on a new line by hyphenating.
- `nextWordFromIndex:forward:` (page 266)
  Returns the index of the first character of the word after or before the given index.

Calculating Ranges

- `itemNumberInTextList:atIndex:` (page 265)
  Returns the range of the item at the given index within the given list.
- `rangeOfTextBlock:atIndex:` (page 266)
  Returns the range of the individual text block that contains the given location.
- `rangeOfTextList:atIndex:` (page 267)
  Returns the range of the given text list that contains the given location.
- `rangeOfTextTable:atIndex:` (page 267)
  Returns the range of the given text table that contains the given location

Generating Data

- `dataFromRange:documentAttributes:error:` (page 256)
  Returns an `NSData` object that contains a text stream corresponding to the characters and attributes within the given range.
- `fileWrapperFromRange:documentAttributes:error:` (page 259)
  Returns an `NSFileWrapper` object that contains a text stream corresponding to the characters and attributes within the given range.
- `docFormatFromRange:documentAttributes:` (page 257)
  Returns an `NSData` object that contains a Microsoft Word–format stream corresponding to the characters and attributes within the specified range.
- `RTFFromRange:documentAttributes:` (page 269)
  Returns an `NSData` object that contains an RTF stream corresponding to the characters and attributes within the given range, omitting all attachment attributes.
- `RTFDFromRange:documentAttributes:` (page 268)
  Returns an NSData object that contains an RTFD stream corresponding to the characters and attributes within *aRange*.
- `RTFDFileWrapperFromRange:documentAttributes:` (page 268)
  Returns an `NSFileWrapper` object that contains an RTFD document corresponding to the characters and attributes within the given range.

Drawing the String

- `drawAtPoint:` (page 257)
  Draws the receiver with its font and other display attributes at the given point in the currently focused `NSView`.
- `drawInRect:` (page 258)
  Draws the receiver with its font and other display attributes within the given rectangle in the currently focused `NSView`, clipping the text layout to this rectangle.
- `drawWithRect:options:` (page 259)
  Draws the receiver with the specified options, within the given rectangle in the current graphics context.
- `size` (page 270)
  Returns the bounding box of the marks that the receiver draws.

Getting the Bounding Rectangle of Rendered Strings

`- boundingRectWithSize:options:` (page 255)
Calculates and returns bounding rectangle for the receiver drawn using the options specified, within the given rectangle in the current graphics context.

Testing String Data Sources

`+ textTypes` (page 254)
Returns an array of UTI strings identifying the file types supported by the receiver, either directly or through a user-installed filter service.

`+ textUnfilteredTypes` (page 255)
Returns an array of UTI strings identifying the file types supported directly by the receiver.

Deprecated Methods

`+ textFileTypes` (page 253) Deprecated in Mac OS X v10.5
Returns an array of strings representing those file types that can be loaded as text. (Deprecated. Use `textTypes` (page 254) instead.)

`+ textPasteboardTypes` (page 253) Deprecated in Mac OS X v10.5
Returns an array of pasteboard types that can be loaded as text. (Deprecated. Use `textTypes` (page 254) instead.)

`+ textUnfilteredFileTypes` (page 254) Deprecated in Mac OS X v10.5
Returns an array of strings representing those file types that can be loaded as a text. (Deprecated. Use `textUnfilteredTypes` (page 255) instead.)

`+ textUnfilteredPasteboardTypes` (page 255) Deprecated in Mac OS X v10.5
Returns an array of pasteboard types that can be loaded as text. (Deprecated. Use `textUnfilteredTypes` (page 255) instead.)

Class Methods

attributedStringWithAttachment:

Creates an attributed string with an attachment.

```
+ (NSAttributedString *)attributedStringWithAttachment:(NSTextAttachment *)attachment
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CoreRecipes

Declared In
`NSTextAttachment.h`

textFileTypes

Returns an array of strings representing those file types that can be loaded as text. (Deprecated in Mac OS X v10.5. Use `textTypes` (page 254) instead.)

```
+ (NSArray *)textFileTypes
```

Discussion
This list includes all file types supported by text classes, plus those types that can be converted to supported file types through a user-installed filter service. The array returned by this method may be passed directly to *NSOpenPanel* method `runModalForTypes:` (page 1821).

File types are identified by extension and HFS file types. By default, the list returned by this method includes "`txt`", "`rtf`", "`rtfd`", and "`html`".

When creating a subclass of `NSAttributedString` that accepts text data from non-default file types, override `textUnfilteredFileTypes` (page 254) to notify `NSAttributedString` of the file types your class supports.

Availability
Available in Mac OS X v10.1 and later.
Deprecated in Mac OS X v10.5.

See Also
`+ textUnfilteredFileTypes` (page 254)

Declared In
`NSAttributedString.h`

textPasteboardTypes

Returns an array of pasteboard types that can be loaded as text. (Deprecated in Mac OS X v10.5. Use `textTypes` (page 254) instead.)

```
+ (NSArray *)textPasteboardTypes
```

Discussion
This list includes all pasteboard types supported by text classes and those that can be converted to supported pasteboard types through a user-installed filter service.

By default, the list returned by this method includes `NSHTMLPboardType`, `NSRTFPboardType`, `NSRTFDPboardType`, and `NSStringPboardType`.

When creating a subclass of `NSAttributedString` that accepts text data from non-default pasteboard types, override `textUnfilteredPasteboardTypes` (page 255) to notify `NSAttributedString` of the pasteboard types your class supports.

Availability
Available in Mac OS X v10.1 and later.
Deprecated in Mac OS X v10.5.

See Also
`+ textUnfilteredPasteboardTypes` (page 255)

Related Sample Code
GLUT

Declared In
NSAttributedString.h

textTypes

Returns an array of UTI strings identifying the file types supported by the receiver, either directly or through a user-installed filter service.

```
+ (NSArray *)textTypes
```

Return Value
An array of `NSString` objects, each of which contains a UTI identifying a supported file type.

Discussion
The returned list includes UTIs all file types supported by the receiver plus those that can be opened by the receiver after being converted by a user-installed filter service. You can use the returned UTI strings with any method that supports UTIs.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSAttributedString.h

textUnfilteredFileTypes

Returns an array of strings representing those file types that can be loaded as a text. (Deprecated in Mac OS X v10.5. Use `textUnfilteredTypes` (page 255) instead.)

```
+ (NSArray *)textUnfilteredFileTypes
```

Discussion
This list consists of all file types supported by text classes, but does not include those types that can be converted to supported file types through a user-installed filter service. The array returned by this method may be passed directly to `NSOpenPanel` method `runModalForTypes:` (page 1821).

Availability
Available in Mac OS X v10.1 and later.
Deprecated in Mac OS X v10.5.

See Also
`+ textFileTypes` (page 253)

Declared In
NSAttributedString.h

textUnfilteredPasteboardTypes

Returns an array of pasteboard types that can be loaded as text. (Deprecated in Mac OS X v10.5. Use `textUnfilteredTypes` (page 255) instead.)

```
+ (NSArray *)textUnfilteredPasteboardTypes
```

Discussion
This list consists of all pasteboard types supported by text classes, but does not include those that can be converted to supported pasteboard types through a user-installed filter service.

Availability
Available in Mac OS X v10.1 and later.
Deprecated in Mac OS X v10.5.

See Also
`+ textPasteboardTypes` (page 253)

Declared In
`NSAttributedString.h`

textUnfilteredTypes

Returns an array of UTI strings identifying the file types supported directly by the receiver.

```
+ (NSArray *)textUnfilteredTypes
```

Return Value
An array of `NSString` objects, each of which contains a UTI identifying a supported file type.

Discussion
The returned list includes UTI strings only for those file types that are supported directly by the receiver. It does not include types that are supported through user-installed filter services. You can use the returned UTI strings with any method that supports UTIs.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSAttributedString.h`

Instance Methods

boundingRectWithSize:options:

Calculates and returns bounding rectangle for the receiver drawn using the options specified, within the given rectangle in the current graphics context.

```
- (NSRect)boundingRectWithSize:(NSSize)size options:(NSStringDrawingOptions)options
```

Discussion
The origin of the rectangle returned from this method is the first glyph origin.

The values of `NSStringDrawingOptions` are listed in the `String Drawing Options` (page 2585) section of NSString Additions.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- drawInRect:` (page 258)

Declared In
`NSStringDrawing.h`

containsAttachments

Returns `YES` if the receiver contains any attachment attributes, `NO` otherwise.

```
- (BOOL)containsAttachments
```

Discussion
This method checks only for attachment attributes, not for `NSAttachmentCharacter`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSAttributedString.h`

dataFromRange:documentAttributes:error:

Returns an `NSData` object that contains a text stream corresponding to the characters and attributes within the given range.

```
- (NSData *)dataFromRange:(NSRange)range documentAttributes:(NSDictionary *)dict
    error:(NSError **)error
```

Discussion
Requires a document attributes dictionary *`dict`* specifying at least the `NSDocumentTypeDocumentAttribute` to determine the format to write. Raises an `NSRangeException` if any part of *`range`* lies beyond the end of the receiver's characters. If unsuccessful, returns `nil` after setting *`error`* to point to an `NSError` object that encapsulates the reason why the object could not be created.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- fileWrapperFromRange:documentAttributes:error:` (page 259)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSAttributedString.h`

docFormatFromRange:documentAttributes:

Returns an `NSData` object that contains a Microsoft Word–format stream corresponding to the characters and attributes within the specified range.

```
- (NSData *)docFormatFromRange:(NSRange)range documentAttributes:(NSDictionary
    *)dict
```

Discussion
The range is passed in the *range* parameter. Also writes the document-level attributes in *dict*, as explained in "Constants" (page 271). If there are no document-level attributes, *dict* can be `nil`. Raises an `NSRangeException` if any part of *range* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSAttributedString.h`

doubleClickAtIndex:

Returns the range of characters that form a word (or other linguistic unit) surrounding the given index, taking language characteristics into account.

```
- (NSRange)doubleClickAtIndex:(NSUInteger)index
```

Discussion
Raises an `NSRangeException` if *index* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- nextWordFromIndex:forward:` (page 266)

Declared In
`NSAttributedString.h`

drawAtPoint:

Draws the receiver with its font and other display attributes at the given point in the currently focused `NSView`.

```
- (void)drawAtPoint:(NSPoint)point
```

Discussion
The width (height for vertical layout) of the rendering area is unlimited, unlike `drawInRect:` (page 258), which uses a bounding rectangle. As a result, this method renders the text in a single line.

Don't invoke this method while no `NSView` is focused.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lockFocus` (page 3187) (NSView)
- `size` (page 270)
- `drawInRect:` (page 258)

Related Sample Code
Aperture Edit Plugin - Borders & Titles
CocoaVideoFrameToGWorld
ComplexBrowser
FunHouse
Movie Overlay

Declared In
`NSStringDrawing.h`

drawInRect:

Draws the receiver with its font and other display attributes within the given rectangle in the currently focused `NSView`, clipping the text layout to this rectangle.

```
- (void)drawInRect:(NSRect)rect
```

Discussion
Text is drawn within *`rect`* according to its line sweep direction; for example, Arabic text will begin at the right edge and potentially be clipped on the left.

The *`rect`* parameter determines how many glyphs are typeset within the width of a line, but it's possible for a portion of a glyph to appear outside the area of *`rect`* if the image bounding box of the particular glyph exceeds its typographic bounding box.

If the focus view is flipped, the text origin is set at the upper-left corner of the drawing bounding box; otherwise the origin is set at the lower-left corner. For text rendering, whether the view coordinates are flipped or not doesn't affect the flow of line layout, which goes from top to bottom. However, it affects the interpretation of the text origin. So, for example, if the *`rect`* argument is `{0.0, 0.0, 100.0, 100.0}`, the text origin is {0.0, 0.0} when the view coordinates are flipped and {0.0, 100.0} when not.

Don't invoke this method while no `NSView` is focused.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lockFocus` (page 3187) (NSView)
- `drawAtPoint:` (page 257)

Related Sample Code
CIAnnotation
IBFragmentView
iChatTheater
PhotoSearch

Declared In
NSStringDrawing.h

drawWithRect:options:

Draws the receiver with the specified options, within the given rectangle in the current graphics context.

```
- (void)drawWithRect:(NSRect)rect options:(NSStringDrawingOptions)options
```

Discussion
The *rect* argument's origin field specifies the rendering origin. The point is interpreted as the baseline origin by default. With NSStringDrawingUsesLineFragmentOrigin, it is interpreted as the upper left corner of the line fragment rect. The size field specifies the text container size. The width part of the size field specifies the maximum line fragment width if larger than 0.0. The height defines the maximum size that can be occupied with text if larger than 0.0 and NSStringDrawingUsesLineFragmentOrigin is specified. If NSStringDrawingUsesLineFragmentOrigin is not specified, height is ignored and considered to be single-line rendering (NSLineBreakByWordWrapping and NSLineBreakByCharWrapping are treated as NSLineBreakByClipping).

The values of NSStringDrawingOptions are listed in the String Drawing Options (page 2585) section of NSString Additions.

You should only invoke this method when there is a current graphics context.

Availability
Available in Mac OS X v10.4 and later.

See Also
- drawAtPoint: (page 257) (NSView)
- lockFocus (page 3187)

Declared In
NSStringDrawing.h

fileWrapperFromRange:documentAttributes:error:

Returns an NSFileWrapper object that contains a text stream corresponding to the characters and attributes within the given range.

```
- (NSFileWrapper *)fileWrapperFromRange:(NSRange)range
    documentAttributes:(NSDictionary *)dict error:(NSError **)error
```

Discussion
Requires a document attributes dictionary *dict* specifying at least the NSDocumentTypeDocumentAttribute to determine the format to write. Raises an NSRangeException if any part of *range* lies beyond the end of the receiver's characters. Returns a directory file wrapper for those document types for which it is appropriate; otherwise a regular file wrapper. If unsuccessful, returns nil after setting *error* to point to an NSError object that encapsulates the reason why the object could not be created.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `dataFromRange:documentAttributes:error:` (page 256)

Declared In
`NSAttributedString.h`

fontAttributesInRange:

Returns the font attributes in effect for the character at the given location.

```
- (NSDictionary *)fontAttributesInRange:(NSRange)aRange
```

Discussion
Returns the font attributes in effect for the character at *aRange*.*location*. Font attributes are all those listed in "Standard Attributes" (page 271), except `NSLinkAttributeName`, `NSParagraphStyleAttributeName`, and `NSAttachmentAttributeName`. Use this method to obtain font attributes that are to be copied or pasted with "copy font" operations. Raises an `NSRangeException` if any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rulerAttributesInRange:` (page 269)

Declared In
`NSAttributedString.h`

initWithData:options:documentAttributes:error:

Initializes and returns a new `NSAttributedString` object from the data contained in the given `NSData` object.

```
- (id)initWithData:(NSData *)data options:(NSDictionary *)options
    documentAttributes:(NSDictionary **)dict error:(NSError **)error
```

Discussion
The *options* dictionary can contain the values described in "Option keys for importing documents" (page 285) to specify how the document should be loaded. If `NSDocumentTypeDocumentOption` is specified, the document is treated as being in the specified format. If `NSDocumentTypeDocumentOption` is not specified, the method examines the document and loads it using whatever format it seems to contain. Also returns by reference in *dict* a dictionary containing document-level attributes described in "Constants" (page 271). The *dict* parameter may be nil, in which case no document attributes are returned. Returns `nil` if *data* can't be decoded, after setting *error* to point to an NSError that encapsulates the reason why the attributed string object could not be created.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
CoreRecipes

Declared In
NSAttributedString.h

initWithDocFormat:documentAttributes:

Initializes and returns a new NSAttributedString object from Microsoft Word format data contained in the given NSData object.

```
- (id)initWithDocFormat:(NSData *)data documentAttributes:(NSDictionary
    **)docAttributes
```

Discussion
Also returns by reference in *docAttributes* a dictionary containing document-level attributes described in "Document Attributes" (page 279). *docAttributes* may be NULL, in which case no document attributes are returned. Returns nil if *data* can't be decoded.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSAttributedString.h

initWithHTML:baseURL:documentAttributes:

Initializes and returns a new NSAttributedString object from the HTML contained in the given object and base URL.

```
- (id)initWithHTML:(NSData *)data baseURL:(NSURL *)aURL
    documentAttributes:(NSDictionary **)docAttributes
```

Discussion
Also returns by reference in *docAttributes* a dictionary containing document-level attributes described in "Document Attributes" (page 279). *docAttributes* may be NULL, in which case no document attributes are returned. Returns an initialized object, or nil if the file at *aURL* can't be decoded.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSAttributedString.h

initWithHTML:documentAttributes:

Initializes and returns a new NSAttributedString object from HTML contained in the given data object.

```
- (id)initWithHTML:(NSData *)data documentAttributes:(NSDictionary **)docAttributes
```

Discussion
Also returns by reference in *docAttributes* a dictionary containing document-level attributes described in "Document Attributes" (page 279). *docAttributes* may be NULL, in which case no document attributes are returned. Returns nil if *data* can't be decoded.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ObjectPath

Declared In
`NSAttributedString.h`

initWithHTML:options:documentAttributes:

Initializes and returns a new `NSAttributedString` object from HTML contained in the given data object.

```
- (id)initWithHTML:(NSData *)data options:(NSDictionary *)options
    documentAttributes:(NSDictionary **)dict
```

Discussion
The *options* dictionary can contain the values described in "Option keys for importing documents" (page 285).

Also returns by reference in *docAttributes* a dictionary containing document-level attributes described in "Document Attributes" (page 279). *docAttributes* may be `NULL`, in which case no document attributes are returned. Returns `nil` if *data* can't be decoded.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSAttributedString.h`

initWithPath:documentAttributes:

Initializes a new `NSAttributedString` object from RTF or RTFD data contained in the file at the given path.

```
- (id)initWithPath:(NSString *)path documentAttributes:(NSDictionary **)docAttributes
```

Discussion
The contents of *path* will be examined to best load the file in whatever format it's in. Filter services can be used to convert the file into a format recognized by Cocoa. Also returns by reference in *docAttributes* a dictionary containing document-level attributes described in "Document Attributes" (page 279). *docAttributes* may be `NULL`, in which case no document attributes are returned. Returns an initialized object, or `nil` if the file at *path* can't be decoded.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DemoAssistant
FancyAbout
iSpend
VertexPerformanceTest

Declared In
NSAttributedString.h

initWithRTF:documentAttributes:

Initializes a new NSAttributedString object by decoding the stream of RTF commands and data contained in the given data object.

```
- (id)initWithRTF:(NSData *)rtfData documentAttributes:(NSDictionary **)docAttributes
```

Discussion
Also returns by reference in *docAttributes* a dictionary containing document-level attributes described in "Document Attributes" (page 279). *docAttributes* may be NULL, in which case no document attributes are returned. Returns an initialized object, or nil if *rtfData* can't be decoded.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CoreRecipes
iSpend
Spotlight

Declared In
NSAttributedString.h

initWithRTFD:documentAttributes:

Initializes a new NSAttributedString object by decoding the stream of RTFD commands and data contained in the given data object.

```
- (id)initWithRTFD:(NSData *)rtfdData documentAttributes:(NSDictionary
    **)docAttributes
```

Discussion
Also returns by reference in *docAttributes* a dictionary containing document-level attributes described in "Constants" (page 271). *docAttributes* may be NULL, in which case no document attributes are returned. Returns an initialized object, or nil if *rtfData* can't be decoded.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSAttributedString.h

initWithRTFDFileWrapper:documentAttributes:

Initializes a new NSAttributedString object from the given NSFileWrapper object containing an RTFD document.

```
- (id)initWithRTFDFileWrapper:(NSFileWrapper *)wrapper
    documentAttributes:(NSDictionary **)docAttributes
```

Discussion
Also returns by reference in *docAttributes* a dictionary containing document-level attributes described in "Document Attributes" (page 279). *docAttributes* may be NULL, in which case no document attributes are returned. Returns an initialized object, or nil if *wrapper* can't be interpreted as an RTFD document.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSAttributedString.h

initWithURL:documentAttributes:

Initializes a new NSAttributedString object from the data at the given URL.

```
- (id)initWithURL:(NSURL *)aURL documentAttributes:(NSDictionary **)docAttributes
```

Discussion
The contents of *aURL* are examined to best load the file in whatever format it's in. Filter services can be used to convert the file into a format recognized by Cocoa. Also returns by reference in *docAttributes* a dictionary containing document-level attributes described in "Document Attributes" (page 279). *docAttributes* may be NULL, in which case no document attributes are returned. Returns an initialized object, or nil if the file at *path* can't be decoded.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSAttributedString.h

initWithURL:options:documentAttributes:error:

Initializes a new NSAttributedString object from the contents of the given URL.

```
- (id)initWithURL:(NSURL *)url options:(NSDictionary *)options
    documentAttributes:(NSDictionary **)dict error:(NSError **)error
```

Discussion
Filter services can be used to convert the file into a format recognized by Cocoa. The *options* dictionary specifies how the document should be loaded and can contain the values described in "Option keys for importing documents" (page 285).

If NSDocumentTypeDocumentOption is specified, the document is treated as being in the specified format. If NSDocumentTypeDocumentOption is not specified, the method examines the document and loads it using whatever format it seems to contain.

Also returns by reference in *dict* a dictionary containing document-level attributes described in "Document Attributes" (page 279). The *dict* parameter may be nil, in which case no document attributes are returned. Returns an initialized object, or nil if the file at *url* can't be decoded, after setting *error* to point to an NSError object that encapsulates the reason why the attributed string object could not be created.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
CoreTextRTF
TextSizingExample

Declared In
NSAttributedString.h

itemNumberInTextList:atIndex:

Returns the range of the item at the given index within the given list.

```
- (NSInteger)itemNumberInTextList:(NSTextList *)list atIndex:(NSUInteger)location
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- rangeOfTextBlock:atIndex: (page 266)
- rangeOfTextList:atIndex: (page 267)
- rangeOfTextTable:atIndex: (page 267)

Declared In
NSAttributedString.h

lineBreakBeforeIndex:withinRange:

Returns the index of the closest character before the given index, and within the given range, that can be placed on a new line when laying out text.

```
- (NSUInteger)lineBreakBeforeIndex:(NSUInteger)index withinRange:(NSRange)aRange
```

Discussion
In other words, finds the appropriate line break when the character at *index* won't fit on the same line as the character at the beginning of *aRange*. Returns NSNotFound if no line break is possible before *index*.Raises an NSRangeException if *index* or any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- lineBreakByHyphenatingBeforeIndex:withinRange: (page 265)

Declared In
NSAttributedString.h

lineBreakByHyphenatingBeforeIndex:withinRange:

Returns the index of the closest character before the given index, and within the given range, that can be placed on a new line by hyphenating.

```
- (NSUInteger)lineBreakByHyphenatingBeforeIndex:(NSUInteger)location
    withinRange:(NSRange)aRange
```

Discussion
In other words, during text layout, finds the appropriate line break by hyphenation (the character index at which the hyphen glyph should be inserted) when the character at `index` won't fit on the same line as the character at the beginning of `aRange`. Returns `NSNotFound` if no line break by hyphenation is possible before `index`.Raises an `NSRangeException` if `index` or any part of `aRange` lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- lineBreakBeforeIndex:withinRange:` (page 265)

Declared In
`NSAttributedString.h`

nextWordFromIndex:forward:

Returns the index of the first character of the word after or before the given index.

```
- (NSUInteger)nextWordFromIndex:(NSUInteger)index forward:(BOOL)flag
```

Discussion
If `flag` is `YES`, this is the first character after `index` that begins a word; if `flag` is `NO`, it's the first character before `index` that begins a word, whether `index` is located within a word or not. If `index` lies at either end of the string and the search direction would progress past that end, it's returned unchanged. This method is intended for moving the insertion point during editing, not for linguistic analysis or parsing of text.Raises an `NSRangeException` if `index` lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- lineBreakBeforeIndex:withinRange:` (page 265)

Declared In
`NSAttributedString.h`

rangeOfTextBlock:atIndex:

Returns the range of the individual text block that contains the given location.

```
- (NSRange)rangeOfTextBlock:(NSTextBlock *)block atIndex:(NSUInteger)location
```

Discussion
The individual text is given by `block` and contains `location`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `itemNumberInTextList:atIndex:` (page 265)
- `rangeOfTextList:atIndex:` (page 267)
- `rangeOfTextTable:atIndex:` (page 267)

Related Sample Code
iSpend

Declared In
`NSAttributedString.h`

rangeOfTextList:atIndex:

Returns the range of the given text list that contains the given location.

```
- (NSRange)rangeOfTextList:(NSTextList *)list atIndex:(NSUInteger)location
```

Discussion
Returns the range of the *list* that contains *location*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `itemNumberInTextList:atIndex:` (page 265)
- `rangeOfTextBlock:atIndex:` (page 266)
- `rangeOfTextTable:atIndex:` (page 267)

Declared In
`NSAttributedString.h`

rangeOfTextTable:atIndex:

Returns the range of the given text table that contains the given location

```
- (NSRange)rangeOfTextTable:(NSTextTable *)table atIndex:(NSUInteger)location
```

Discussion
Returns the range of the text *table* that contains *location*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `itemNumberInTextList:atIndex:` (page 265)
- `rangeOfTextList:atIndex:` (page 267)
- `rangeOfTextBlock:atIndex:` (page 266)

Related Sample Code
iSpend

Declared In
NSAttributedString.h

RTFDFileWrapperFromRange:documentAttributes:

Returns an NSFileWrapper object that contains an RTFD document corresponding to the characters and attributes within the given range.

```
- (NSFileWrapper *)RTFDFileWrapperFromRange:(NSRange)aRange
    documentAttributes:(NSDictionary *)docAttributes
```

Discussion
The file wrapper also includes the document-level attributes in *docAttributes*, as explained in RTF Files and Attributed Strings. If there are no document-level attributes, *docAttributes* can be nil. Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.You can save the file wrapper using the NSFileWrapper method writeToFile:atomically:updateFilenames: (page 1137).

Availability
Available in Mac OS X v10.0 and later.

See Also
- RTFFromRange:documentAttributes: (page 269)
- RTFDFromRange:documentAttributes: (page 268)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSAttributedString.h

RTFDFromRange:documentAttributes:

Returns an NSData object that contains an RTFD stream corresponding to the characters and attributes within *aRange*.

```
- (NSData *)RTFDFromRange:(NSRange)aRange documentAttributes:(NSDictionary
    *)docAttributes
```

Discussion
Also writes the document-level attributes in *docAttributes*, as explained in RTF Files and Attributed Strings. If there are no document-level attributes, *docAttributes* can be nil. Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

When writing data to the pasteboard, you can use the NSData object as the first argument to the NSPasteboard method setData:forType: (page 1898), with a second argument of NSRTFDPboardType.

Availability
Available in Mac OS X v10.0 and later.

See Also
- RTFFromRange:documentAttributes: (page 269)
- RTFDFileWrapperFromRange:documentAttributes: (page 268)

Related Sample Code
DemoAssistant
GLUT

Declared In
NSAttributedString.h

RTFFromRange:documentAttributes:

Returns an NSData object that contains an RTF stream corresponding to the characters and attributes within the given range, omitting all attachment attributes.

```
- (NSData *)RTFFromRange:(NSRange)aRange documentAttributes:(NSDictionary
    *)docAttributes
```

Discussion
Also writes the document-level attributes in *docAttributes*, as explained in RTF Files and Attributed Strings. If there are no document-level attributes, *docAttributes* can be nil. Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.When writing data to the pasteboard, you can use the NSData object as the first argument to the NSPasteboard method setData:forType: (page 1898), with a second argument of NSRTFPboardType. Although this method strips attachments, it leaves the attachment characters in the text itself. The NSText method RTFFromRange: (page 2732), on the other hand, does strip attachment characters when extracting RTF.

Availability
Available in Mac OS X v10.0 and later.

See Also
- RTFDFromRange:documentAttributes: (page 268)
- RTFDFileWrapperFromRange:documentAttributes: (page 268)

Related Sample Code
CoreRecipes
VertexPerformanceTest

Declared In
NSAttributedString.h

rulerAttributesInRange:

Returns the ruler (paragraph) attributes in effect for the characters within the given range.

```
- (NSDictionary *)rulerAttributesInRange:(NSRange)aRange
```

Discussion
The only ruler attribute currently defined is that named by NSParagraphStyleAttributeName. Use this method to obtain attributes that are to be copied or pasted with "copy ruler" operations. Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `fontAttributesInRange:` (page 260)

Declared In
`NSAttributedString.h`

size

Returns the bounding box of the marks that the receiver draws.

```
- (NSSize)size
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawAtPoint:` (page 257)
- `drawInRect:` (page 258)

Related Sample Code
Aperture Edit Plugin - Borders & Titles
CIAnnotation
ComplexBrowser
IBFragmentView
PhotoSearch

Declared In
`NSStringDrawing.h`

URLAtIndex:effectiveRange:

Returns a URL, either from a link attribute or from text at the given location that appears to be a URL string, for use in automatic link detection.

```
- (NSURL *)URLAtIndex:(NSUInteger)location
    effectiveRange:(NSRangePointer)effectiveRange
```

Parameters

location
The character index in the string at which the method checks for a link.

effectiveRange
The actual range covered by the link attribute or URL string, or of non-URL text if no apparent URL is found.

Return Value
The URL found at *location*.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSAttributedString.h`

Constants

Standard Attributes

Attributed strings support the following standard attributes for text. If the key is not in the dictionary, then use the default values described below.

```
NSString *NSFontAttributeName;
NSString *NSParagraphStyleAttributeName;
NSString *NSForegroundColorAttributeName;
NSString *NSUnderlineStyleAttributeName;
NSString *NSSuperscriptAttributeName;
NSString *NSBackgroundColorAttributeName;
NSString *NSAttachmentAttributeName;
NSString *NSLigatureAttributeName;
NSString *NSBaselineOffsetAttributeName;
NSString *NSKernAttributeName;
NSString *NSLinkAttributeName;
NSString *NSStrokeWidthAttributeName;
NSString *NSStrokeColorAttributeName;
NSString *NSUnderlineColorAttributeName;
NSString *NSStrikethroughStyleAttributeName;
NSString *NSStrikethroughColorAttributeName;
NSString *NSShadowAttributeName;
NSString *NSObliquenessAttributeName;
NSString *NSExpansionAttributeName;
NSString *NSCursorAttributeName;
NSString *NSToolTipAttributeName;
NSString *NSMarkedClauseSegmentAttributeName;
NSString *NSWritingDirectionAttributeName;
```

Constants

`NSFontAttributeName`
`NSFont`

Default Helvetica 12-point

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSParagraphStyleAttributeName`
`NSParagraphStyle`

Default as returned by the `NSParagraphStyle` method `defaultParagraphStyle`

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSForegroundColorAttributeName`
`NSColor`

Default `blackColor`

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSUnderlineStyleAttributeName`
`NSNumber` containing integer

Default `0`, no underline. See "`Underlining Patterns`" (page 275), "`Underlining Styles`" (page 274), and "`Underline Masks`" (page 276) for mask values.

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSSuperscriptAttributeName`
`NSNumber` containing integer

Default `0`

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSBackgroundColorAttributeName`
`NSColor`

Default `nil`, no background

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSAttachmentAttributeName`
`NSTextAttachment`

Default `nil`, no attachment

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSLigatureAttributeName`
`NSNumber` containing integer

Default `1`, standard ligatures; `0`, no ligatures; `2`, all ligatures

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSBaselineOffsetAttributeName`
`NSNumber` containing floating point value, as points offset from baseline

Default `0.0`

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSKernAttributeName`
`NSNumber` containing floating point value, as points by which to modify default kerning

Default `nil`, use default kerning specified in font file; `0.0`, kerning off; non-zero, points by which to modify default kerning

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSLinkAttributeName`
`NSURL` (preferred) or `NSString`

Default `nil`, no link

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSStrokeWidthAttributeName`
`NSNumber` containing floating point value, as percent of font point size

Default `0`, no stroke; positive, stroke alone; negative, stroke and fill (a typical value for outlined text would be `3.0`)

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSStrokeColorAttributeName`
`NSColor`

Default `nil`, same as foreground color

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSUnderlineColorAttributeName`
`NSColor`

Default `nil`, same as foreground color

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSStrikethroughStyleAttributeName`
`NSNumber` containing integer

Default `0`, no strikethrough. See "Underlining Patterns" (page 275), "Underlining Styles" (page 274), and "Underline Masks" (page 276) for mask values.

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSStrikethroughColorAttributeName`
`NSColor`

Default `nil`, same as foreground color

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSShadowAttributeName`
`NSShadow`

Default `nil`, no shadow

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSObliquenessAttributeName`
`NSNumber` containing floating point value, as skew to be applied to glyphs

Default `0.0`, no skew

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSExpansionAttributeName`
`NSNumber` containing floating point value, as log of expansion factor to be applied to glyphs

Default `0.0`, no expansion

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSCursorAttributeName`
`NSCursor`

Default as returned by the `NSCursor` method `IBeamCursor`

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSToolTipAttributeName`
`NSString`

Default `nil`, no tooltip

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSMarkedClauseSegmentAttributeName`
`NSNumber` containing an integer, as an index in marked text indicating clause segments

Available in Mac OS X v10.5 and later.

Declared in `NSAttributedString.h`.

`NSWritingDirectionAttributeName`
An `NSArray` of `NSNumbers`.

This provides a means to override the default bidi algorithm, equivalent to the use of bidi control characters LRE, RLE, LRO, or RLO paired with PDF, as a higher-level attribute. This is the NSAttributedString equivalent of HTML's dir attribute and/or BDO element. The array represents nested embeddings or overrides, in order from outermost to innermost. The values of the NSNumbers should be 0, 1, 2, or 3, for LRE, RLE, LRO, or RLO respectively; these should be regarded as `NSWritingDirectionLeftToRight` (page 2748) or `NSWritingDirectionRightToLeft` (page 2748) plus `NSTextWritingDirectionEmbedding` or `NSTextWritingDirectionOverride`.

Available in Mac OS X v10.6 and later.

Declared in `NSAttributedString.h`.

Underlining Styles

These constants define underlining style values for `NSUnderlineStyleAttributeName` (page 272) and `NSStrikethroughStyleAttributeName` (page 273).

```
enum {
    NSUnderlineStyleNone   = 0x00,
    NSUnderlineStyleSingle   = 0x01,
    NSUnderlineStyleThick   = 0x02,
    NSUnderlineStyleDouble   = 0x09
};
```

Constants

`NSUnderlineStyleNone`
Do not draw an underline.

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSUnderlineStyleSingle`
Draw an underline consisting of a single line.

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSUnderlineStyleThick`
Draw an underline consisting of a thick line.

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSUnderlineStyleDouble`
Draw an underline consisting of a double line.

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

Discussion
See also "Underline Masks" (page 276) and "Underlining Patterns" (page 275). The style, pattern, and optionally by-word mask are OR'd together to produce the value for `NSUnderlineStyleAttributeName` (page 272) and `NSStrikethroughStyleAttributeName` (page 273).

Declared In
`NSAttributedString.h`

Underlining Patterns

These constants define underlining pattern values for `NSUnderlineStyleAttributeName` (page 272) and `NSStrikethroughStyleAttributeName` (page 273).

```
enum {
    NSUnderlinePatternSolid   = 0x0000,
    NSUnderlinePatternDot   = 0x0100,
    NSUnderlinePatternDash   = 0x0200,
    NSUnderlinePatternDashDot   = 0x0300,
    NSUnderlinePatternDashDotDot  = 0x0400
};
```

Constants
`NSUnderlinePatternSolid`
Draw a solid underline.

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSUnderlinePatternDot`
Draw an underline using a pattern of dots.

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSUnderlinePatternDash`
Draw an underline using a pattern of dashes.

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSUnderlinePatternDashDot`
Draw an underline using a pattern of alternating dashes and dots.

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

`NSUnderlinePatternDashDotDot`
Draw an underline using a pattern of a dash followed by two dots.

Available in Mac OS X v10.3 and later.

Declared in `NSAttributedString.h`.

Discussion
See also "`Underline Masks`" (page 276) and "`Underlining Styles`" (page 274). The style, pattern, and optionally by-word mask are OR'd together to produce the value for `NSUnderlineStyleAttributeName` (page 272) and `NSStrikethroughStyleAttributeName` (page 273).

Deprecated Underlining Styles

Deprecated constants previously used for underline style. (Deprecated. See "Underlining Patterns" (page 275) for the correct replacements.)

```
enum {
    NSNoUnderlineStyle = 0,
    NSSingleUnderlineStyle
};
NSUInteger NSUnderlineStrikethroughMask;
```

Constants
`NSNoUnderlineStyle`
Deprecated see "Underlining Patterns" (page 275) for the correct replacements.

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

`NSSingleUnderlineStyle`
Deprecated see "Underlining Patterns" (page 275) for the correct replacements.

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

Underline Masks

This constant defines the underlining style for `NSUnderlineStyleAttributeName` (page 272) and `NSStrikethroughStyleAttributeName` (page 273).

```
unsigned NSUnderlineByWordMask;
```

Constants
`NSUnderlineByWordMask`
Draw the underline only underneath words, not underneath whitespace.

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

Discussion
Use this constant with the desired underline style to create the given effect. For example, to get a thick underline only underneath words, set `NSUnderlineStyleAttribute` to `(NSUnderlineStyleThick | NSUnderlineByWordMask)`. Also see "Underlining Styles" (page 274) and "Underlining Patterns" (page 275).

Declared In
`NSAttributedString.h`

Glyph Info Attribute

This object provides a means to override the standard glyph generation.

```
NSString *NSGlyphInfoAttributeName;
```

Constants
`NSGlyphInfoAttributeName`
The name of an `NSGlyphInfo` object.

`NSLayoutManager` assigns the glyph specified by this glyph info to the entire attribute range, provided that its contents match the specified base string, and that the specified glyph is available in the font specified by `NSFontAttributeName`.

Available in Mac OS X v10.2 and later.

Declared in `NSAttributedString.h`.

Declared In
`NSAttributedString.h`

Character Shape Attribute

The character shape feature type (`kCharacterShapeType`) is used when a single font contains different appearances for the same character shape, and these shapes are not traditionally treated as swashes. It is needed for languages such as Chinese that have both traditional and simplified character sets.

```
NSString *NSCharacterShapeAttributeName;
```

Constants
`NSCharacterShapeAttributeName`
An integer value. The value is interpreted as Apple Type Services `kCharacterShapeType selector + 1`.

The default value is 0 (disable). 1 is `kTraditionalCharactersSelector`, and so on. Refer to `<ATS/SFNTLayoutTypes.h>` and Font Features in *ATSUI Programming Guide* for additional information.

Available in Mac OS X v10.0 and later.

Declared in `NSAttributedString.h`.

Declared In
`NSAttributedString.h`

Document Types

The following values can be returned for the `NSDocumentTypeDocumentAttribute` (page 281) key in the document attributes dictionary.

```
NSString *NSPlainTextDocumentType;
NSString *NSRTFTextDocumentType;
NSString *NSRTFDTextDocumentType;
NSString *NSMacSimpleTextDocumentType;
NSString *NSHTMLTextDocumentType;
NSString *NSDocFormatTextDocumentType;
NSString *NSWordMLTextDocumentType;
NSString *NSWebArchiveTextDocumentType;
NSString *NSOfficeOpenXMLTextDocumentType;
NSString *NSOpenDocumentTextDocumentType;
```

Constants

`NSPlainTextDocumentType`
Plain text document.
Available in Mac OS X v10.0 and later.
Declared in `NSAttributedString.h`.

`NSRTFTextDocumentType`
Rich text format document.
Available in Mac OS X v10.0 and later.
Declared in `NSAttributedString.h`.

`NSRTFDTextDocumentType`
Rich text format with attachments document.
Available in Mac OS X v10.0 and later.
Declared in `NSAttributedString.h`.

`NSMacSimpleTextDocumentType`
Macintosh SimpleText document.
Available in Mac OS X v10.0 and later.
Declared in `NSAttributedString.h`.

`NSHTMLTextDocumentType`
Hypertext Markup Language (HTML) document.
Available in Mac OS X v10.0 and later.
Declared in `NSAttributedString.h`.

`NSDocFormatTextDocumentType`
Microsoft Word document.
Available in Mac OS X v10.3 and later.
Declared in `NSAttributedString.h`.

`NSWordMLTextDocumentType`
Microsoft Word XML (WordML schema) document.
Available in Mac OS X v10.3 and later.
Declared in `NSAttributedString.h`.

`NSWebArchiveTextDocumentType`
Web Kit WebArchive document.
Available in Mac OS X v10.4 and later.
Declared in `NSAttributedString.h`.

`NSOfficeOpenXMLTextDocumentType`
ECMA Office Open XML text document format.

Available in Mac OS X v10.5 and later.

Declared in `NSAttributedString.h`.

`NSOpenDocumentTextDocumentType`
OASIS Open Document text document format.

Available in Mac OS X v10.5 and later.

Declared in `NSAttributedString.h`.

Discussion
See also NSDocumentTypeDocumentOption (page 285).

Declared In
`NSAttributedString.h`

Document Attributes

The `init...` methods can return a dictionary with the following document-wide attributes (attribute identifiers available on Mac OS X v10.4 and later; use actual string value keys for earlier systems):

```
NSString *NSAuthorDocumentAttribute;
NSString *NSBackgroundColorDocumentAttribute;
NSString *NSBottomMarginDocumentAttribute;
NSString *NSCharacterEncodingDocumentAttribute;
NSString *NSCategoryDocumentAttribute;
NSString *NSCocoaVersionDocumentAttribute;
NSString *NSCommentDocumentAttribute;
NSString *NSCompanyDocumentAttribute;
NSString *NSConvertedDocumentAttribute;
NSString *NSCopyrightDocumentAttribute;
NSString *NSCreationTimeDocumentAttribute;
NSString *NSDefaultTabIntervalDocumentAttribute;
NSString *NSDocumentTypeDocumentAttribute;
NSString *NSEditorDocumentAttribute;
NSString *NSFileTypeDocumentAttribute;
NSString *NSFileTypeDocumentOption;
NSString *NSHyphenationFactorDocumentAttribute;
NSString *NSKeywordsDocumentAttribute;
NSString *NSLeftMarginDocumentAttribute;
NSString *NSManagerDocumentAttribute;
NSString *NSModificationTimeDocumentAttribute;
NSString *NSPaperSizeDocumentAttribute;
NSString *NSReadOnlyDocumentAttribute;
NSString *NSRightMarginDocumentAttribute;
NSString *NSSubjectDocumentAttribute;
NSString *NSTitleDocumentAttribute;
NSString *NSTopMarginDocumentAttribute;
NSString *NSViewModeDocumentAttribute;
NSString *NSViewSizeDocumentAttribute;
NSString *NSViewZoomDocumentAttribute;
```

Constants

`NSPaperSizeDocumentAttribute`

NSValue, containing NSSize.

Mac OS X v10.3 and earlier string constant is `@"PaperSize"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSLeftMarginDocumentAttribute`

NSNumber, containing a float, in points.

Mac OS X v10.3 and earlier string constant is `@"LeftMargin"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSRightMarginDocumentAttribute`

NSNumber, containing a float, in points.

Mac OS X v10.3 and earlier string constant is `@"RightMargin"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSTopMarginDocumentAttribute`
NSNumber, containing a float, in points.

Mac OS X v10.3 and earlier string constant is `@"TopMargin"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSBottomMarginDocumentAttribute`
NSNumber, containing a float, in points.

Mac OS X v10.3 and earlier string constant is `@"BottomMargin"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSHyphenationFactorDocumentAttribute`
NSNumber, containing a float; 0 = off, 1 = full hyphenation.

Mac OS X v10.3 and earlier string constant is `@"HyphenationFactor"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSDocumentTypeDocumentAttribute`
How the document was interpreted; one of the values in "Document Types" (page 277).

Mac OS X v10.3 and earlier string constant is `@"DocumentType"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSCharacterEncodingDocumentAttribute`
NSNumber, containing an int specifying the `NSStringEncoding` for the file; for reading and writing plain text files and writing HTML; default for plain text is the default encoding; default for HTML is UTF-8.

Mac OS X v10.3 and earlier string constant is `@"CharacterEncoding"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSViewSizeDocumentAttribute`
NSValue, containing NSSize.

Mac OS X v10.3 and earlier string constant is `@"ViewSize"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSViewZoomDocumentAttribute`
Mac OS X v10.3 and earlier string constant is `@"ViewZoom"`.

NSValue, containing a float; 100 = 100% zoom.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSViewModeDocumentAttribute`
NSValue, containing an int; 0 = normal; 1 = page layout (use value of `@"PaperSize"`).

Mac OS X v10.3 and earlier string constant is `@"ViewMode"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSBackgroundColorDocumentAttribute`

NSColor, representing the document-wide page background color.

Mac OS X v10.3 and earlier string constant is `@"BackgroundColor"`.

For applications linked on versions prior to Mac OS X v10.5, HTML import sets the `NSBackgroundColorDocumentAttribute` to `[NSColor whiteColor]` in cases in which the HTML does not specify a background color. For applications linked on Mac OS X v10.5 and later, no `NSBackgroundColorDocumentAttribute` is set in these cases.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSCocoaVersionDocumentAttribute`

NSNumber, containing a float. For RTF files only, stores the version of Cocoa with which the file was created. Absence of this value indicates RTF file not created by Cocoa or its predecessors.

Values less than 100 are pre–Mac OS X; 100 is Mac OS X v10.0 or v10.1; 102 is Mac OS X v10.2 and 10.3; values greater than 102 correspond to values of `NSAppKitVersionNumber` on Mac OS X v10.4 and later.

Mac OS X v10.3 and earlier string constant is `@"CocoaRTFVersion"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSReadOnlyDocumentAttribute`

NSNumber, containing int. If missing or 0 or negative, not read only; 1 or more, read only.

Note that this has nothing to do with the file system protection on the file, but instead can affect how the file should be displayed to the user.

Mac OS X v10.3 and earlier string constant is `@"ReadOnly"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSConvertedDocumentAttribute`

NSNumber, containing an int. Indicates whether the file was converted by a filter service.

If missing or 0, the file was originally in the format specified by document type. If negative, the file was originally in the format specified by document type, but the conversion to NSAttributedString may have been lossy. If 1 or more, it was converted to this type by a filter service.

Mac OS X v10.3 and earlier string constant is `@"Converted"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSDefaultTabIntervalDocumentAttribute`

NSNumber containing a float. Represents the document-wide default tab stop interval.

Mac OS X v10.3 and earlier string constant is `@"DefaultTabInterval"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSTitleDocumentAttribute`

NSString containing document title.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSCompanyDocumentAttribute`
NSString containing company or organization name.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSCopyrightDocumentAttribute`
NSString containing document copyright info.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSSubjectDocumentAttribute`
NSString containing subject of document.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSAuthorDocumentAttribute`
`NSString` containing author name.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSKeywordsDocumentAttribute`
`NSArray` of `NSString`, containing keywords.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSCommentDocumentAttribute`
`NSString` containing document comments.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSEditorDocumentAttribute`
`NSString` containing name of person who last edited the document.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSCreationTimeDocumentAttribute`
`NSDate` containing the creation date of the document; note that this is not the file system creation date of the file, but of the document.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSModificationTimeDocumentAttribute`
`NSDate` containing the modification date of the document contents.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSManagerDocumentAttribute`
`NSString` containing the name of the author's manager.

Available in Mac OS X v10.6 and later.

Declared in `NSAttributedString.h`.

`NSCategoryDocumentAttribute`
`NSString` containing the document's category.

Available in Mac OS X v10.6 and later.

Declared in `NSAttributedString.h`.

`NSFileTypeDocumentAttribute`
`NSString` indicating which document type was used to interpret the document, specified as a UTI; for reading, this is available along with `NSDocumentTypeDocumentAttribute` (page 281), but for writing the two are mutually exclusive.

Available in Mac OS X v10.6 and later.

Declared in `NSAttributedString.h`.

`NSFileTypeDocumentOption`
`NSString` indicating a document type to be forced when loading the document, specified as a UTI string; mutually exclusive with `NSDocumentTypeDocumentOption` (page 285).

Available in Mac OS X v10.6 and later.

Declared in `NSAttributedString.h`.

Attributes for generating HTML

These document-wide attributes provide control over the form of generated HTML—you use them only for writing HTML

```
NSString *NSExcludedElementsDocumentAttribute;
NSString *NSTextEncodingNameDocumentAttribute;
NSString *NSPrefixSpacesDocumentAttribute;
```

Constants

`NSExcludedElementsDocumentAttribute`
An `NSArray` object containing `NSString` objects, representing HTML elements not to be used in generated HTML.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSTextEncodingNameDocumentAttribute`
An `NSString` object containing the name, IANA or otherwise, of a text encoding to be used; mutually exclusive with `NSCharacterEncodingDocumentAttribute`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSPrefixSpacesDocumentAttribute`
An `NSNumber` containing an integer (default 0) representing the number of spaces per level by which to indent certain nested HTML elements.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

Discussion

`NSExcludedElementsDocumentAttribute` allows control over the tags used. The recognized values in the `NSExcludedElementsDocumentAttribute` array are (case-insensitive) HTML tags, plus DOCTYPE (representing a doctype declaration) and XML (representing an XML declaration). By default, if this attribute is not present, the excluded elements will be those deprecated in HTML 4 (APPLET, BASEFONT, CENTER, DIR, FONT, ISINDEX, MENU, S, STRIKE, and U) plus XML. If XML is on the list, HTML forms are used; if XML is not

on the list, XHTML forms are used where there is a distinction. Either `NSCharacterEncodingDocumentAttribute` or `NSTextEncodingNameDocumentAttribute` may be used to control the encoding used for generated HTML; character entities are used for characters not representable in the specified encoding. `NSPrefixSpacesDocumentAttribute` allows some control over formatting.

Declared In
`NSAttributedString.h`

Option keys for importing documents

These option keys are recognized for importing documents using initWithData:options:documentAttributes:error: (page 260), initWithHTML:options:documentAttributes: (page 262), initWithURL:options:documentAttributes:error: (page 264), or the `readFrom...` methods (such as `readFromData:options:documentAttributes:` (page 1710)) implemented by `NSMutableAttributedString`.

```
NSString *NSBaseURLDocumentOption;
NSString *NSCharacterEncodingDocumentOption;
NSString *NSDefaultAttributesDocumentOption;
NSString *NSDocumentTypeDocumentOption;
NSString *NSTextEncodingNameDocumentOption;
NSString *NSTextSizeMultiplierDocumentOption;
NSString *NSTimeoutDocumentOption;
NSString *NSWebPreferencesDocumentOption;
NSString *NSWebResourceLoadDelegateDocumentOption;
```

Constants

`NSCharacterEncodingDocumentOption`
For plain text documents; NSNumber containing the unsigned int `NSStringEncoding` to override any encoding specified in an HTML document. Previous string constant was `@"CharacterEncoding"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSBaseURLDocumentOption`
For HTML documents; `NSURL` containing base URL. Previous string constant was `@"BaseURL"`

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSDefaultAttributesDocumentOption`
For plain text documents; NSDictionary containing attributes to be applied to plain files. Previous string constant was `@"DefaultAttributes"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSDocumentTypeDocumentOption`
One of the document types described in "Document Types" (page 277), indicating a document type to be forced when loading the document. Previous string constant was `@"DocumentType"`.

Available in Mac OS X v10.4 and later.

Declared in `NSAttributedString.h`.

`NSTextEncodingNameDocumentOption`
: NSString containing the name, IANA or otherwise, of a text encoding to override any encoding specified in an HTML document. Mutually exclusive with `@"CharacterEncoding"`. Previous string constant was `@"TextEncodingName"`.

  Available in Mac OS X v10.4 and later.

  Declared in `NSAttributedString.h`.

`NSTimeoutDocumentOption`
: NSNumber containing float. Time in seconds to wait for a document to finish loading. Previous string constant was `@"Timeout"`.

  Available in Mac OS X v10.4 and later.

  Declared in `NSAttributedString.h`.

`NSWebPreferencesDocumentOption`
: WebPreferences; for HTML only, specifies a WebPreferences object. If not present, a default set of preferences is used. Previous string constant was `@"WebPreferences"`.

  Available in Mac OS X v10.4 and later.

  Declared in `NSAttributedString.h`.

`NSWebResourceLoadDelegateDocumentOption`
: `NSObject`; for HTML only, specifies an object to serve as the web resource loading delegate.

  If not present, a default delegate is used that permits the loading of subsidiary resources but does not respond to authentication challenges. Previous string constant was `@"WebResourceLoadDelegate"`.

  Available in Mac OS X v10.4 and later.

  Declared in `NSAttributedString.h`.

`NSTextSizeMultiplierDocumentOption`
: Specifies a scale factor for font sizes.. `NSNumber` containing float, default 1.0; for HTML only, corresponding to WebView's `textSizeMultiplier`.

  There is no textual equivalent for Mac OS X v10.3.

  Available in Mac OS X v10.4 and later.

  Declared in `NSAttributedString.h`.

Discussion
In Mac OS X v10.3, the options key `@"UseWebKit"` specifies that WebKit–based HTML importing be used (and must be specified for the other options to be recognized). In Mac OS X v10.4 and later, WebKit is always used for HTML documents, and all of the options except `@"UseWebKit"` are recognized (attribute identifiers are available on Mac OS X v10.4 and later; use actual string value keys for Mac OS X v10.3):

NSSpellingStateAttributeName

This constant is used as, and recognized only as, a temporary attribute. It indicates that spelling and/or grammar indicators should be shown for the specified characters.

```
NSString *NSSpellingStateAttributeName;
```

Constants

`NSSpellingStateAttributeName`

An integer value. Defaults to 0, indicating no grammar or spelling error. See "NSSpellingStateAttributeName Flags" (page 287) for possible values.

This key is available in Mac OS X v10.2 and later, but its interpretation changed in Mac OS X v10.5. Previously, any non-zero value caused the spelling indicator to be displayed. For Mac OS X v10.5 and later, the (integer) value is treated as being composed of the spelling and grammar flags. See "NSSpellingStateAttributeName Flags" (page 287) for possible values.

Available in Mac OS X v10.5 and later.

Declared in `NSAttributedString.h`.

NSSpellingStateAttributeName Flags

These constants control the display of the spelling and grammar indicators on text, highlighting portions of the text that are flagged for spelling or grammar issues. These regions are denoted by a temporary attribute on the layout manager, using the `NSSpellingStateAttributeName` (page 287) key.

```
enum {
    NSSpellingStateSpellingFlag = (1 << 0),
    NSSpellingStateGrammarFlag  = (1 << 1)
};
```

Constants

`NSSpellingStateSpellingFlag`

Flag for spelling issues.

Available in Mac OS X v10.5 and later.

Declared in `NSAttributedString.h`.

`NSSpellingStateGrammarFlag`

Flag for grammar issues.

Available in Mac OS X v10.5 and later.

Declared in `NSAttributedString.h`.

NSBezierPath Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSBezierPath.h
Companion guide	Cocoa Drawing Guide
Related sample code	DockTile Sketch-112 SpeedometerView WebKitPluginStarter WebKitPluginWithJavaScript

Overview

An `NSBezierPath` object allows you to create paths using PostScript-style commands. Paths consist of straight and curved line segments joined together. Paths can form recognizable shapes such as rectangles, ovals, arcs, and glyphs; they can also form complex polygons using either straight or curved line segments. A single path can be closed by connecting its two endpoints, or it can be left open.

An `NSBezierPath` object can contain multiple disconnected paths, whether they are closed or open. Each of these paths is referred to as a subpath. The subpaths of an `NSBezierPath` object must be manipulated as a group. The only way to manipulate subpaths individually is to create separate `NSBezierPath` objects for each.

For a given `NSBezierPath` object, you can stroke the path's outline or fill the region occupied by the path. You can also use the path as a clipping region for views or other regions. Using methods of `NSBezierPath`, you can also perform hit detection on the filled or stroked path. Hit detection is needed to implement interactive graphics, as in rubberbanding and dragging operations.

The current graphics context is automatically saved and restored for all drawing operations involving `NSBezierPath` objects, so your application does not need to worry about the graphics settings changing across invocations.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

NSCopying
- `copyWithZone:`

Tasks

Creating an NSBezierPath Object

`+ bezierPath` (page 294)
Creates and returns a new `NSBezierPath` object.

`+ bezierPathWithOvalInRect:` (page 295)
Creates and returns a new `NSBezierPath` object initialized with an oval path inscribed in the specified rectangle.

`+ bezierPathWithRect:` (page 295)
Creates and returns a new `NSBezierPath` object initialized with a rectangular path.

`+ bezierPathWithRoundedRect:xRadius:yRadius:` (page 296)
Creates and returns a new `NSBezierPath` object initialized with a rounded rectangular path.

`- bezierPathByFlatteningPath` (page 313)
Creates and returns a “flattened” copy of the receiver.

`- bezierPathByReversingPath` (page 314)
Creates and returns a new `NSBezierPath` object with the reversed contents of the receiver’s path.

Constructing Paths

`- moveToPoint:` (page 324)
Moves the receiver’s current point to the specified location.

`- lineToPoint:` (page 322)
Appends a straight line to the receiver’s path

`- curveToPoint:controlPoint1:controlPoint2:` (page 317)
Adds a Bezier cubic curve to the receiver’s path.

`- closePath` (page 315)
Closes the most recently added subpath.

`- relativeMoveToPoint:` (page 326)
Moves the receiver’s current point to a new point whose location is the specified distance from the current point.

- `relativeLineToPoint:` (page 325)
  Appends a straight line segment to the receiver's path starting at the current point and moving towards the specified point, relative to the current location.
- `relativeCurveToPoint:controlPoint1:controlPoint2:` (page 325)
  Adds a Bezier cubic curve to the receiver's path from the current point to a new location, which is specified as a relative distance from the current point.

Appending Common Shapes to a Path

- `appendBezierPath:` (page 307)
  Appends the contents of the specified path object to the receiver's path.
- `appendBezierPathWithPoints:count:` (page 312)
  Appends a series of line segments to the receiver's path.
- `appendBezierPathWithOvalInRect:` (page 311)
  Appends an oval path to the receiver, inscribing the oval in the specified rectangle.
- `appendBezierPathWithArcFromPoint:toPoint:radius:` (page 307)
  Appends an arc to the receiver's path.
- `appendBezierPathWithArcWithCenter:radius:startAngle:endAngle:` (page 308)
  Appends an arc of a circle to the receiver's path.
- `appendBezierPathWithArcWithCenter:radius:startAngle:endAngle:clockwise:` (page 309)
  Appends an arc of a circle to the receiver's path.
- `appendBezierPathWithGlyph:inFont:` (page 309)
  Appends an outline of the specified glyph to the receiver's path.
- `appendBezierPathWithGlyphs:count:inFont:` (page 310)
  Appends the outlines of the specified glyphs to the receiver's path.
- `appendBezierPathWithPackedGlyphs:` (page 311)
  Appends an array of packed glyphs to the receiver's path.
- `appendBezierPathWithRect:` (page 312)
  Appends a rectangular path to the receiver's path.
- `appendBezierPathWithRoundedRect:xRadius:yRadius:` (page 313)
  Appends a rounded rectangular path to the receiver's path.

Accessing Path Attributes

+ `defaultWindingRule` (page 299)
  Returns the default winding rule used to fill all paths.
+ `setDefaultWindingRule:` (page 304)
  Sets the default winding rule used to fill all paths.
- `windingRule` (page 334)
  Returns the winding rule used to fill the receiver's path.
- `setWindingRule:` (page 332)
  Sets the winding rule used to fill the receiver's path.
+ `defaultLineCapStyle` (page 298)
  Returns the default line cap style for all paths.

+ `setDefaultLineCapStyle:` (page 302)
  Sets the default line cap style for all paths.
- `lineCapStyle` (page 322)
  Returns the line cap style for the receiver's path.
- `setLineCapStyle:` (page 329)
  Sets the line cap style for the receiver's path.
+ `defaultLineJoinStyle` (page 298)
  Returns the default line join style for all paths.
+ `setDefaultLineJoinStyle:` (page 302)
  Sets the default line join style for all paths.
- `lineJoinStyle` (page 322)
  Returns the receiver's line join style.
- `setLineJoinStyle:` (page 330)
  Sets the line join style for the receiver's path.
+ `defaultLineWidth` (page 299)
  Returns the default line width for the all paths.
+ `setDefaultLineWidth:` (page 303)
  Sets the default line width for all paths.
- `lineWidth` (page 323)
  Returns the line width of the receiver's path.
- `setLineWidth:` (page 331)
  Sets the line width of the receiver's path.
+ `defaultMiterLimit` (page 299)
  Returns the default miter limit for all paths.
+ `setDefaultMiterLimit:` (page 304)
  Sets the default miter limit for all paths.
- `miterLimit` (page 323)
  Returns the miter limit of the receiver's path.
- `setMiterLimit:` (page 332)
  Sets the miter limit for the receiver's path.
+ `defaultFlatness` (page 297)
  Returns the default flatness value for all paths.
+ `setDefaultFlatness:` (page 301)
  Sets the default flatness value for all paths.
- `flatness` (page 320)
  Returns the flatness value of the receiver's path.
- `setFlatness:` (page 329)
  Sets the flatness value for the receiver's path.
- `getLineDash:count:phase:` (page 321)
  Returns the line-stroking pattern for the receiver.
- `setLineDash:count:phase:` (page 330)
  Sets the line-stroking pattern for the receiver.

Drawing Paths

- `stroke` (page 333)
  Draws a line along the receiver's path using the current stroke color and drawing attributes.
- `fill` (page 320)
  Paints the region enclosed by the receiver's path.
+ `fillRect:` (page 300)
  Fills the specified rectangular path with the current fill color.
+ `strokeRect:` (page 306)
  Strokes the path of the specified rectangle using the current stroke color and the default drawing attributes.
+ `strokeLineFromPoint:toPoint:` (page 305)
  Strokes a line between two points using the current stroke color and the default drawing attributes.
+ `drawPackedGlyphs:atPoint:` (page 300)
  Draws a set of packed glyphs at the specified point in the current coordinate system.

Clipping Paths

- `addClip` (page 306)
  Intersects the area enclosed by the receiver's path with the clipping path of the current graphics context and makes the resulting shape the current clipping path.
- `setClip` (page 328)
  Replaces the clipping path of the current graphics context with the area inside the receiver's path.
+ `clipRect:` (page 297)
  Intersects the specified rectangle with the clipping path of the current graphics context and makes the resulting shape the current clipping path

Hit Detection

- `containsPoint:` (page 316)
  Returns a Boolean value indicating whether the receiver contains the specified point.

Querying Paths

- `bounds` (page 314)
  Returns the bounding box of the receiver's path.
- `controlPointBounds` (page 316)
  Returns the bounding box of the receiver's path, including any control points.
- `currentPoint` (page 317)
  Returns the receiver's current point (the trailing point or ending point in the most recently added segment).
- `isEmpty` (page 321)
  Returns a Boolean value indicating whether the receiver is empty.

Applying Transformations

- `transformUsingAffineTransform:` (page 333)
  Transforms all points in the receiver using the specified transform.

Accessing Elements of a Path

- `elementCount` (page 319)
  Returns the total number of path elements in the receiver's path.
- `elementAtIndex:` (page 318)
  Returns the type of path element at the specified index.
- `elementAtIndex:associatedPoints:` (page 318)
  Gets the element type and (and optionally) the associated points for the path element at the specified index.
- `removeAllPoints` (page 326)
  Removes all path elements from the receiver, effectively clearing the path.
- `setAssociatedPoints:atIndex:` (page 327)
  Changes the points associated with the specified path element.

Caching Paths

- `cachesBezierPath` (page 315) Deprecated in Mac OS X v10.0
  Returns a Boolean value indicating whether this object maintains a cached image of its path.
- `setCachesBezierPath:` (page 328) Deprecated in Mac OS X v10.0
  Sets whether the receiver should cache its path information.

Class Methods

bezierPath

Creates and returns a new `NSBezierPath` object.

```
+ (NSBezierPath *)bezierPath
```

Return Value
A new empty path object.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DockTile
Sketch-112
SpeedometerView
WebKitPluginStarter

WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

bezierPathWithOvalInRect:

Creates and returns a new NSBezierPath object initialized with an oval path inscribed in the specified rectangle.

```
+ (NSBezierPath *)bezierPathWithOvalInRect:(NSRect)aRect
```

Parameters

aRect

The rectangle in which to inscribe an oval.

Return Value
An NSBezierPath new path object with the oval path.

Discussion
If the *aRect* parameter specifies a square, the inscribed path is a circle. The path is constructed by starting in the lower-right quadrant of the rectangle and adding arc segments counterclockwise to complete the oval.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ bezierPath (page 294)
- appendBezierPathWithOvalInRect: (page 311)

Related Sample Code
BezierPathLab
BindingsJoystick
MenuItemView
Sketch-112
Worm

Declared In
NSBezierPath.h

bezierPathWithRect:

Creates and returns a new NSBezierPath object initialized with a rectangular path.

```
+ (NSBezierPath *)bezierPathWithRect:(NSRect)aRect
```

Parameters

aRect

The rectangle describing the path to create.

Return Value
A new path object with the rectangular path.

Discussion
The path is constructed by starting at the origin of *aRect* and adding line segments in a counterclockwise direction.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ bezierPath (page 294)
- appendBezierPathWithRect: (page 312)
+ fillRect: (page 300)
+ strokeRect: (page 306)

Related Sample Code
BezierPathLab
Cropped Image
Link Snoop
PDFKitLinker2
Sketch-112

Declared In
NSBezierPath.h

bezierPathWithRoundedRect:xRadius:yRadius:

Creates and returns a new NSBezierPath object initialized with a rounded rectangular path.

```
+ (NSBezierPath *)bezierPathWithRoundedRect:(NSRect)rect xRadius:(CGFloat)xRadius
     yRadius:(CGFloat)yRadius
```

Parameters

rect
The rectangle that defines the basic shape of the path.

xRadius
The radius of each corner oval along the x-axis. Values larger than half the rectangle's width are clamped to half the width.

yRadius
The radius of each corner oval along the y-axis. Values larger than half the rectangle's height are clamped to half the height.

Return Value
A new path object with the rounded rectangular path.

Discussion
The path is constructed in a counter-clockwise direction, starting at the top-left corner of the rectangle. If either one of the radius parameters contains the value 0.0, the returned path is a plain rectangle without rounded corners.

Availability
Available in Mac OS X v10.5 and later.

See Also
+ bezierPath (page 294)
- appendBezierPathWithRoundedRect:xRadius:yRadius: (page 313)

Related Sample Code
AnimatedTableView
CocoaSlides
TrackBall

Declared In
NSBezierPath.h

clipRect:

Intersects the specified rectangle with the clipping path of the current graphics context and makes the resulting shape the current clipping path

```
+ (void)clipRect:(NSRect)aRect
```

Parameters
aRect
The rectangle to intersect with the current clipping path.

Availability
Available in Mac OS X v10.0 and later.

See Also
- addClip (page 306)
- setClip (page 328)

Related Sample Code
QuickLookSketch
Sketch+Accessibility
Sketch-112
Transformed Image

Declared In
NSBezierPath.h

defaultFlatness

Returns the default flatness value for all paths.

```
+ (CGFloat)defaultFlatness
```

Return Value
The default value for determining the smoothness of curved paths, or 0.6 if no other value has been set.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ setDefaultFlatness: (page 301)
- flatness (page 320)

Declared In
NSBezierPath.h

defaultLineCapStyle

Returns the default line cap style for all paths.

```
+ (NSLineCapStyle)defaultLineCapStyle
```

Return Value
The default line cap style or NSButtLineCapStyle if no other style has been set. For a list of values, see "Constants" (page 335).

Discussion
The default line cap style can be overridden for individual paths by setting a custom style for that path using the setLineCapStyle: (page 329) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ setDefaultLineCapStyle: (page 302)
+ defaultLineJoinStyle (page 298)
+ defaultLineWidth (page 299)
- lineCapStyle (page 322)

Declared In
NSBezierPath.h

defaultLineJoinStyle

Returns the default line join style for all paths.

```
+ (NSLineJoinStyle)defaultLineJoinStyle
```

Return Value
The default line join style or NSMiterLineJoinStyle if no other value has been set. For a list of values, see "Constants" (page 335).

Availability
Available in Mac OS X v10.0 and later.

See Also
+ setDefaultLineJoinStyle: (page 302)
+ defaultLineCapStyle (page 298)
+ defaultLineWidth (page 299)
- lineJoinStyle (page 322)

Declared In
NSBezierPath.h

defaultLineWidth

Returns the default line width for the all paths.

```
+ (CGFloat)defaultLineWidth
```

Return Value
The default line width, measured in points in the user coordinate space, or 1.0 if no other value has been set.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ setDefaultLineWidth: (page 303)
+ defaultLineCapStyle (page 298)
+ defaultLineJoinStyle (page 298)
- lineWidth (page 323)

Declared In
NSBezierPath.h

defaultMiterLimit

Returns the default miter limit for all paths.

```
+ (CGFloat)defaultMiterLimit
```

Return Value
The default miter limit for all paths, or 10.0 if no other value has been set.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ setDefaultMiterLimit: (page 304)
- miterLimit (page 323)

Declared In
NSBezierPath.h

defaultWindingRule

Returns the default winding rule used to fill all paths.

```
+ (NSWindingRule)defaultWindingRule
```

Return Value
The current default winding rule or `NSNonZeroWindingRule` if no default rule has been set. This value may be either `NSNonZeroWindingRule` or `NSEvenOddWindingRule`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ setDefaultWindingRule:` (page 304)
`- windingRule` (page 334)

Declared In
`NSBezierPath.h`

drawPackedGlyphs:atPoint:

Draws a set of packed glyphs at the specified point in the current coordinate system.

```
+ (void)drawPackedGlyphs:(const char *)packedGlyphs atPoint:(NSPoint)aPoint
```

Parameters
packedGlyphs
A C-style array containing one or more `CGGlyph` data types terminated by a `NULL` character.
aPoint
The starting point at which to draw the glyphs.

Discussion
This method draws the glyphs immediately.

You should avoid using this method directly. Instead, use the `appendBezierPathWithGlyph:inFont:` (page 309) and `appendBezierPathWithGlyphs:count:inFont:` (page 310) methods to create a path with one or more glyphs.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- appendBezierPathWithPackedGlyphs:` (page 311)
`- set` (page 711) (NSColor)

Declared In
`NSBezierPath.h`

fillRect:

Fills the specified rectangular path with the current fill color.

```
+ (void)fillRect:(NSRect)aRect
```

Parameters
aRect
A rectangle in the current coordinate system.

Discussion
This method fills the specified region immediately. This method uses the compositing operation returned by the `compositingOperation` method of `NSGraphicsContext`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- appendBezierPathWithRect:` (page 312)
- `+ bezierPathWithRect:` (page 295)
- `+ strokeRect:` (page 306)
- `- compositingOperation` (page 1304) (NSGraphicsContext)
- `- set` (page 711) (NSColor)

Related Sample Code
DragItemAround
ImageApp
MatrixMixerTest
SampleRaster
WhackedTV

Declared In
`NSBezierPath.h`

setDefaultFlatness:

Sets the default flatness value for all paths.

```
+ (void)setDefaultFlatness:(CGFloat)flatness
```

Parameters
flatness
The default flatness value.

Discussion
The flatness value specifies the accuracy (or smoothness) with which curves are rendered. It is also the maximum error tolerance (measured in pixels) for rendering curves, where smaller numbers give smoother curves at the expense of more computation. The exact interpretation may vary slightly on different rendering devices.

The default flatness value is 0.6, which yields smooth curves.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `+ defaultFlatness` (page 297)
- `- setFlatness:` (page 329)

Declared In
`NSBezierPath.h`

setDefaultLineCapStyle:

Sets the default line cap style for all paths.

```
+ (void)setDefaultLineCapStyle:(NSLineCapStyle)lineCap
```

Parameters

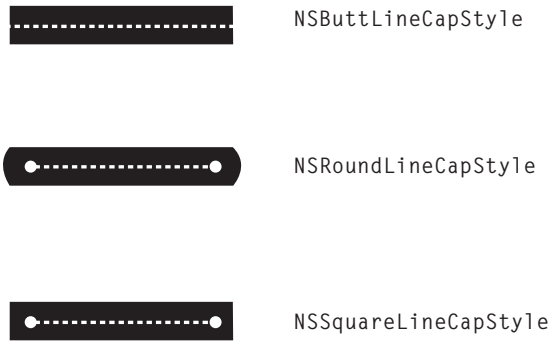
lineCap

The default line cap style. For a list of values, see "Constants" (page 335).

Discussion

The line cap style specifies the shape of the endpoints of an open path when stroked. Figure 13-1 (page 302) shows the appearance of the available line cap styles.

Figure 13-1 Line cap styles



Availability

Available in Mac OS X v10.0 and later.

See Also

+ defaultLineCapStyle (page 298)
+ setDefaultLineJoinStyle: (page 302)
+ setDefaultLineWidth: (page 303)
- setLineCapStyle: (page 329)

Declared In

NSBezierPath.h

setDefaultLineJoinStyle:

Sets the default line join style for all paths.

```
+ (void)setDefaultLineJoinStyle:(NSLineJoinStyle)lineJoinStyle
```

Parameters

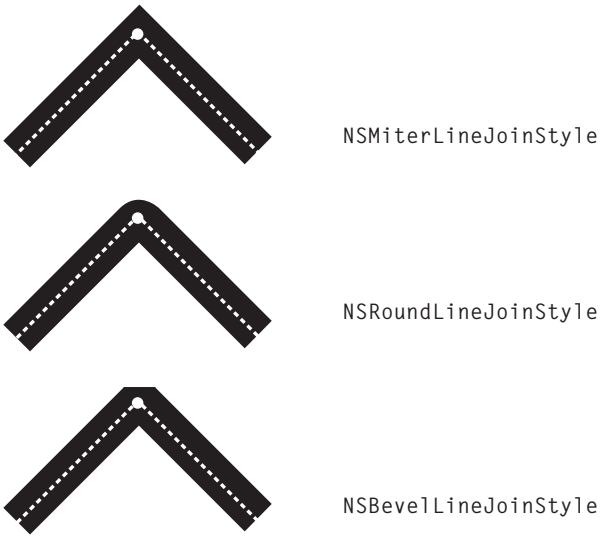
lineJoinStyle

The default line join style. For a list of values, see "Constants" (page 335).

Discussion

The line join style specifies the shape of the joints between connected segments of a stroked path. Figure 13-2 (page 303) shows the appearance of the available line join styles.

Figure 13-2 Line join styles



Availability
Available in Mac OS X v10.0 and later.

See Also
+ defaultLineJoinStyle (page 298)
+ setDefaultLineCapStyle: (page 302)
+ setDefaultLineWidth: (page 303)
+ setDefaultMiterLimit: (page 304)
- setLineJoinStyle: (page 330)

Declared In
NSBezierPath.h

setDefaultLineWidth:

Sets the default line width for all paths.

```
+ (void)setDefaultLineWidth:(CGFloat)width
```

Parameters

width

The default line width, measured in points in the user coordinate space.

Discussion
The line width defines the thickness of stroked paths. A width of 0 is interpreted as the thinnest line that can be rendered on a particular device. The actual rendered line width may vary from the specified width by as much as 2 device pixels, depending on the position of the line with respect to the pixel grid and the current anti-aliasing settings. The width of the line may also be affected by scaling factors specified in the current transformation matrix of the active graphics context.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ defaultLineWidth (page 299)
+ setDefaultLineCapStyle: (page 302)
+ setDefaultLineJoinStyle: (page 302)
- setLineWidth: (page 331)

Related Sample Code
ClockControl
CocoaDragAndDrop

Declared In
NSBezierPath.h

setDefaultMiterLimit:

Sets the default miter limit for all paths.

```
+ (void)setDefaultMiterLimit:(CGFloat)limit
```

Parameters
limit
The default limit at which miter joins are converted to bevel joins.

Discussion
The miter limit helps you avoid spikes at the junction of two line segments connected by a miter join (NSMiterLineJoinStyle). If the ratio of the miter length—the diagonal length of the miter join—to the line thickness exceeds the miter limit, the joint is converted to a bevel join. The default miter limit value is 10, which converts miters whose angle at the joint is less than 11 degrees.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ defaultMiterLimit (page 299)
+ setDefaultLineJoinStyle: (page 302)
- setMiterLimit: (page 332)

Declared In
NSBezierPath.h

setDefaultWindingRule:

Sets the default winding rule used to fill all paths.

```
+ (void)setDefaultWindingRule:(NSWindingRule)windingRule
```

Parameters
windingRule
The winding rule to use if no winding rule is set explicitly for a path object. This value may be either NSNonZeroWindingRule or NSEvenOddWindingRule.

Discussion
Winding rules determine how to paint (or fill) the region enclosed by a path. You use this method to set the default rule that is applied to paths that do not have a custom winding rule assigned.

For more information on how winding rules affect the appearance of filled paths, see "Winding Rules and Filling Paths".

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ defaultWindingRule` (page 299)
`- setWindingRule:` (page 332)

Declared In
`NSBezierPath.h`

strokeLineFromPoint:toPoint:

Strokes a line between two points using the current stroke color and the default drawing attributes.

```
+ (void)strokeLineFromPoint:(NSPoint)point1 toPoint:(NSPoint)point2
```

Parameters
point1
The starting point of the line.
point2
The ending point of the line.

Discussion
This method strokes the specified path immediately.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- lineToPoint:` (page 322)
`- moveToPoint:` (page 324)
`+ setDefaultLineCapStyle:` (page 302)
`+ setDefaultLineWidth:` (page 303)
`- stroke` (page 333)

Related Sample Code
BindingsJoystick
FilterDemo
MatrixMixerTest
Rulers
WhackedTV

Declared In
`NSBezierPath.h`

strokeRect:

Strokes the path of the specified rectangle using the current stroke color and the default drawing attributes.

```
+ (void)strokeRect:(NSRect)aRect
```

Parameters

aRect

A rectangle in the current coordinate system.

Discussion

The path is drawn beginning at the rectangle's origin and proceeding in a counterclockwise direction. This method strokes the specified path immediately.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `appendBezierPathWithRect:` (page 312)
- `bezierPathWithRect:` (page 295)
- `fillRect:` (page 300)
- `setDefaultLineJoinStyle:` (page 302)
- `setDefaultLineWidth:` (page 303)
- `set` (page 711) (NSColor)

Related Sample Code

CocoaDragAndDrop

SampleRaster

Declared In

`NSBezierPath.h`

Instance Methods

addClip

Intersects the area enclosed by the receiver's path with the clipping path of the current graphics context and makes the resulting shape the current clipping path.

```
- (void)addClip
```

Discussion

This method uses the current winding rule to determine the clipping shape of the receiver. This method does not affect the receiver's path.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `clipRect:` (page 297)
- `setClip` (page 328)

Related Sample Code
BindingsJoystick
IBFragmentView
Reducer
WebKitDOMElementPlugIn

Declared In
`NSBezierPath.h`

appendBezierPath:

Appends the contents of the specified path object to the receiver's path.

```
- (void)appendBezierPath:(NSBezierPath *)aPath
```

Parameters

aPath
The path to add to the receiver.

Discussion
This method adds the commands used to create *aPath* to the end of the receiver's path. This method does not explicitly try to connect the subpaths in the two objects, although the operations in *aPath* may still cause that effect.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBezierPath.h`

appendBezierPathWithArcFromPoint:toPoint:radius:

Appends an arc to the receiver's path.

```
- (void)appendBezierPathWithArcFromPoint:(NSPoint)fromPoint toPoint:(NSPoint)toPoint
    radius:(CGFloat)radius
```

Parameters

fromPoint
The middle point of the angle.

toPoint
The end point of the angle.

radius
The radius of the circle inscribed in the angle.

Discussion
The created arc is defined by a circle inscribed inside the angle specified by three points: the current point, the `fromPoint` parameter, and the `toPoint` parameter (in that order). The arc itself lies on the perimeter of the circle, whose radius is specified by the `radius` parameter. The arc is drawn between the two points of the circle that are tangent to the two legs of the angle.

The arc usually does not contain the points in the *fromPoint* and *toPoint* parameters. If the starting point of the arc does not coincide with the current point, a line is drawn between the two points. The starting point of the arc lies on the line defined by the current point and the *fromPoint* parameter.

You must set the path's current point (using the `moveToPoint:` (page 324) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Depending on the length of the arc, this method may add multiple connected curve segments to the path.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
IBFragmentView

Declared In
`NSBezierPath.h`

appendBezierPathWithArcWithCenter:radius:startAngle:endAngle:

Appends an arc of a circle to the receiver's path.

```
- (void)appendBezierPathWithArcWithCenter:(NSPoint)center radius:(CGFloat)radius
    startAngle:(CGFloat)startAngle endAngle:(CGFloat)endAngle
```

Parameters

center
Specifies the center point of the circle used to define the arc.

radius
Specifies the radius of the circle used to define the arc.

startAngle
Specifies the starting angle of the arc, measured in degrees counterclockwise from the x-axis.

endAngle
Specifies the end angle of the arc, measured in degrees counterclockwise from the x-axis.

Discussion
The created arc lies on the perimeter of the circle, between the angles specified by the *startAngle* and *endAngle* parameters. The arc is drawn in a counterclockwise direction. If the receiver's path is empty, this method sets the current point to the beginning of the arc before adding the arc segment. If the receiver's path is not empty, a line is drawn from the current point to the starting point of the arc.

Depending on the length of the arc, this method may add multiple connected curve segments to the path.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

appendBezierPathWithArcWithCenter:radius:startAngle:endAngle:clockwise:

Appends an arc of a circle to the receiver's path.

```
- (void)appendBezierPathWithArcWithCenter:(NSPoint)center radius:(CGFloat)radius
    startAngle:(CGFloat)startAngle endAngle:(CGFloat)endAngle
    clockwise:(BOOL)clockwise
```

Parameters

center
Specifies the center point of the circle used to define the arc.

radius
Specifies the radius of the circle used to define the arc.

startAngle
Specifies the starting angle of the arc, measured in degrees counterclockwise from the x-axis.

endAngle
Specifies the end angle of the arc, measured in degrees counterclockwise from the x-axis.

clockwise
YES if you want the arc to be drawn in a clockwise direction; otherwise NO to draw the arc in a counterclockwise direction.

Discussion
The created arc lies on the perimeter of the circle, between the angles specified by the *startAngle* and *endAngle* parameters. The arc is drawn in the direction indicated by the *clockwise* parameter. If the receiver's path is empty, this method sets the current point to the beginning of the arc before adding the arc segment. If the receiver's path is not empty, a line is drawn from the current point to the starting point of the arc.

Depending on the length of the arc, this method may add multiple connected curve segments to the path.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

appendBezierPathWithGlyph:inFont:

Appends an outline of the specified glyph to the receiver's path.

```
- (void)appendBezierPathWithGlyph:(NSGlyph)aGlyph inFont:(NSFont *)fontObj
```

Parameters

aGlyph

The glyph to add to the path.

fontObj

The font in which the glyph is encoded.

Discussion

If the glyph is not encoded in the font specified by the *fontObj* parameter—that is, the font does not have an entry for the specified glyph—then no path is appended to the receiver.

You must set the path's current point (using the `moveToPoint:` (page 324) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `appendBezierPathWithGlyphs:count:inFont:` (page 310)
- `appendBezierPathWithPackedGlyphs:` (page 311)
+ `drawPackedGlyphs:atPoint:` (page 300)

Declared In

`NSBezierPath.h`

appendBezierPathWithGlyphs:count:inFont:

Appends the outlines of the specified glyphs to the receiver's path.

```
- (void)appendBezierPathWithGlyphs:(NSGlyph *)glyphs count:(NSInteger)count
    inFont:(NSFont *)fontObj
```

Parameters

glyphs

A C-style array of `NSGlyph` data types to add to the path.

count

The number of glyphs in the *glyphs* parameter.

fontObj

The font in which the glyphs are encoded.

Discussion

If the glyphs are not encoded in the font specified by the *fontObj* parameter—that is, the font does not have an entry for one of the specified glyphs—then no path is appended to the receiver.

You must set the path's current point (using the `moveToPoint:` (page 324) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `appendBezierPathWithGlyph:inFont:` (page 309)

- appendBezierPathWithPackedGlyphs: (page 311)
+ drawPackedGlyphs:atPoint: (page 300)

Declared In
NSBezierPath.h

appendBezierPathWithOvalInRect:

Appends an oval path to the receiver, inscribing the oval in the specified rectangle.

```
- (void)appendBezierPathWithOvalInRect:(NSRect)aRect
```

Parameters

aRect

The rectangle in which to inscribe the oval.

Discussion
Before adding the oval, this method moves the current point, which implicitly closes the current subpath. If the *aRect* parameter specifies a square, the inscribed path is a circle. The path is constructed by starting in the lower-right quadrant of the rectangle and adding arc segments counterclockwise to complete the oval.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CompositeLab
Grady
SonOfSillyBalls

Declared In
NSBezierPath.h

appendBezierPathWithPackedGlyphs:

Appends an array of packed glyphs to the receiver's path.

```
- (void)appendBezierPathWithPackedGlyphs:(const char *)packedGlyphs
```

Parameters

packedGlyphs

A C-style array containing one or more CGGlyph data types terminated by a NULL character.

Discussion
You should avoid using this method directly. Instead, use the appendBezierPathWithGlyph:inFont: (page 309) and appendBezierPathWithGlyphs:count:inFont: (page 310) methods to append glyphs to a path.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ drawPackedGlyphs:atPoint: (page 300)

Related Sample Code
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

appendBezierPathWithPoints:count:

Appends a series of line segments to the receiver's path.

```
- (void)appendBezierPathWithPoints:(NSPointArray)points count:(NSInteger)count
```

Parameters

points
A C-style array of NSPoint data types, each of which contains the end point of the next line segment.

count
The number of points in the *points* parameter.

Discussion
This method interprets the points as a set of connected line segments. If the current path contains an open subpath, a line is created from the last point in that subpath to the first point in the points array. If the current path is empty, the first point in the points array is used to set the starting point of the line segments. Subsequent line segments are added using the remaining points in the array.

This method does not close the path that is created. If you wish to create a closed path, you must do so by explicitly invoking the receiver's closePath (page 315) method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSBezierPath.h

appendBezierPathWithRect:

Appends a rectangular path to the receiver's path.

```
- (void)appendBezierPathWithRect:(NSRect)aRect
```

Parameters

aRect
The rectangle describing the path to create.

Discussion
Before adding the rectangle, this method moves the current point to the origin of the rectangle, which implicitly closes the current subpath (if any). The path is constructed by starting at the origin of *aRect* and adding line segments in a counterclockwise direction. The final segment is added using a closePath (page 315) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ bezierPathWithRect:` (page 295)
`+ fillRect:` (page 300)
`+ strokeRect:` (page 306)

Related Sample Code
Cropped Image
IBFragmentView
TrackBall

Declared In
`NSBezierPath.h`

appendBezierPathWithRoundedRect:xRadius:yRadius:

Appends a rounded rectangular path to the receiver's path.

```
- (void)appendBezierPathWithRoundedRect:(NSRect)rect xRadius:(CGFloat)xRadius
    yRadius:(CGFloat)yRadius
```

Parameters

rect
The rectangle that defines the basic shape of the path.

xRadius
The radius of each corner oval along the x-axis. Values larger than half the rectangle's width are clamped to half the width.

yRadius
The radius of each corner oval along the y-axis. Values larger than half the rectangle's height are clamped to half the height.

Discussion
The path is constructed in a counter-clockwise direction, starting at the top-left corner of the rectangle. If either one of the radius parameters contains the value `0.0`, the returned path is a plain rectangle without rounded corners.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ bezierPathWithRoundedRect:xRadius:yRadius:` (page 296)

Declared In
`NSBezierPath.h`

bezierPathByFlatteningPath

Creates and returns a "flattened" copy of the receiver.

```
- (NSBezierPath *)bezierPathByFlatteningPath
```

Return Value
A new path object whose contents are a flattened version of the receiver's path.

Discussion
Flattening a path converts all curved line segments into straight line approximations. The granularity of the approximations is controlled by the path's current flatness value, which is set using the `setDefaultFlatness:` (page 301) method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBezierPath.h`

bezierPathByReversingPath

Creates and returns a new `NSBezierPath` object with the reversed contents of the receiver's path.

```
- (NSBezierPath *)bezierPathByReversingPath
```

Return Value
A new path object whose contents are a reversed version of the receiver's path.

Discussion
Reversing a path does not necessarily change the appearance of the path when rendered. Instead, it changes the direction in which path segments are drawn. For example, reversing the path of a rectangle (whose line segments are normally drawn starting at the origin and proceeding in a counterclockwise direction) causes its line segments to be drawn in a clockwise direction instead. Drawing a reversed path could affect the appearance of a filled pattern, depending on the pattern and the fill rule in use.

This method reverses each whole or partial subpath in the path object individually.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBezierPath.h`

bounds

Returns the bounding box of the receiver's path.

```
- (NSRect)bounds
```

Return Value
The rectangle that encloses the path of the receiver. If the path contains curve segments, the bounding box encloses the curve but may not enclose the control points used to calculate the curve.

Availability
Available in Mac OS X v10.0 and later.

See Also
- controlPointBounds (page 316)

Related Sample Code
Cropped Image
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

cachesBezierPath

Returns a Boolean value indicating whether this object maintains a cached image of its path. (Deprecated in Mac OS X v10.0.)

```
- (BOOL)cachesBezierPath
```

Return Value
YES if the path maintains a cached image; otherwise, NO.

Discussion
Caching of paths currently has no effect, so method always returns NO.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.0.

See Also
- setCachesBezierPath: (page 328)

Declared In
NSBezierPath.h

closePath

Closes the most recently added subpath.

```
- (void)closePath
```

Discussion
This method closes the current subpath by creating a line segment between the first and last points in the subpath. This method subsequently updates the current point to the end of the newly created line segment, which is also the first point in the now closed subpath.

Availability
Available in Mac OS X v10.0 and later.

See Also
- fill (page 320)

Related Sample Code
CompositeLab
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSBezierPath.h`

containsPoint:

Returns a Boolean value indicating whether the receiver contains the specified point.

```
- (BOOL)containsPoint:(NSPoint)aPoint
```

Parameters
aPoint
The point to test against the path, specified in the path object's coordinate system.

Return Value
`YES` if the path's enclosed area contains the specified point; otherwise, `NO`.

Discussion
This method checks the point against the path itself and the area it encloses. When determining hits in the enclosed area, this method uses the non-zero winding rule (`NSNonZeroWindingRule`). It does not take into account the line width used to stroke the path.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Dicey
ImageMap
ImageMapExample

Declared In
`NSBezierPath.h`

controlPointBounds

Returns the bounding box of the receiver’s path, including any control points.

```
- (NSRect)controlPointBounds
```

Return Value
The rectangle that encloses the receiver's path. If the path contains curve segments, the bounding box encloses the control points of the curves as well as the curves themselves.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `bounds` (page 314)

Declared In
`NSBezierPath.h`

currentPoint

Returns the receiver's current point (the trailing point or ending point in the most recently added segment).

```
- (NSPoint)currentPoint
```

Return Value
The point from which the next drawn line or curve segment begins.

Discussion
If the receiver is empty, this method raises `NSGenericException`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `closePath` (page 315)
- `curveToPoint:controlPoint1:controlPoint2:` (page 317)
- `lineToPoint:` (page 322)
- `moveToPoint:` (page 324)

Declared In
`NSBezierPath.h`

curveToPoint:controlPoint1:controlPoint2:

Adds a Bezier cubic curve to the receiver's path.

```
- (void)curveToPoint:(NSPoint)aPoint controlPoint1:(NSPoint)controlPoint1
    controlPoint2:(NSPoint)controlPoint2
```

Parameters
aPoint
The destination point of the curve segment, specified in the current coordinate system

controlPoint1
The point that determines the shape of the curve near the current point.

controlPoint2
The point that determines the shape of the curve near the destination point.

Discussion
You must set the path's current point (using the `moveToPoint:` (page 324) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- closePath` (page 315)
- `- lineToPoint:` (page 322)
- `- relativeCurveToPoint:controlPoint1:controlPoint2:` (page 325)
+ `setDefaultFlatness:` (page 301)

Related Sample Code
CocoaVideoFrameToGWorld
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSBezierPath.h`

elementAtIndex:

Returns the type of path element at the specified index.

```
- (NSBezierPathElement)elementAtIndex:(NSInteger)index
```

Parameters

index
The index of the desired path element.

Return Value
The type of the path element. For a list of constants, see "NSBezierPathElement" (page 335).

Discussion
Path elements describe the commands used to define a path and include basic commands such as moving to a specific point, creating a line segment, creating a curve, or closing the path. The elements are stored in the order of their execution.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- elementCount` (page 319)
- `- elementAtIndex:associatedPoints:` (page 318)
- `- bezierPathByReversingPath` (page 314)

Declared In
`NSBezierPath.h`

elementAtIndex:associatedPoints:

Gets the element type and (and optionally) the associated points for the path element at the specified index.

```
- (NSBezierPathElement)elementAtIndex:(NSInteger)index
    associatedPoints:(NSPointArray)points
```

Parameters

index

The index of the desired path element.

points

On input, a C-style array containing up to three `NSPoint` data types, or `NULL` if you do not want the points. On output, the data points associated with the specified path element.

Return Value

The type of the path element. For a list of constants, see "NSBezierPathElement" (page 335).

Discussion

If you specify a value for the points parameter, your array must be large enough to hold the number of points for the given path element. Move, close path, and line segment commands return one point. Curve operations return three points.

For curve operations, the order of the points is controlPoint1 (*points*[0]), controlPoint2 (*points*[1]), endPoint (*points*[2]).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `elementCount` (page 319)
- `elementAtIndex:` (page 318)

Declared In

`NSBezierPath.h`

elementCount

Returns the total number of path elements in the receiver's path.

```
- (NSInteger)elementCount
```

Return Value

The number of path elements.

Discussion

Each element type corresponds to one of the operations described in "Path Elements".

Availability

Available in Mac OS X v10.0 and later.

See Also

- `elementAtIndex:` (page 318)
- `elementAtIndex:associatedPoints:` (page 318)

Related Sample Code

Cropped Image

Declared In

`NSBezierPath.h`

fill

Paints the region enclosed by the receiver's path.

```
- (void)fill
```

Discussion
This method fills the path using the current fill color and the receiver's current winding rule. If the path contains any open subpaths, this method implicitly closes them before painting the fill region.

The painted region includes the pixels right up to, but not including, the path line itself. For paths with large line widths, this can result in overlap between the fill region and the stroked path (which is itself centered on the path line).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `stroke` (page 333)
- `windingRule` (page 334)
- `set` (page 711) (NSColor)

Related Sample Code
Cropped Image
Dicey
WebKitPluginStarter
WebKitPluginWithJavaScript
Worm

Declared In
`NSBezierPath.h`

flatness

Returns the flatness value of the receiver's path.

```
- (CGFloat)flatness
```

Return Value
The flatness value of the path. If no value is set, this method returns the default flatness value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFlatness:` (page 329)
+ `defaultFlatness` (page 297)

Declared In
`NSBezierPath.h`

getLineDash:count:phase:

Returns the line-stroking pattern for the receiver.

```
- (void)getLineDash:(CGFloat *)pattern count:(NSInteger *)count phase:(CGFloat
    *)phase
```

Parameters

pattern

On input, a C-style array of floating point values, or `nil` if you do not want the pattern values. On output, this array contains the lengths (measured in points) of the line segments and gaps in the pattern. The values in the array alternate, starting with the first line segment length, followed by the first gap length, followed by the second line segment length, and so on.

count

On input, a pointer to an integer or `nil` if you do not want the number of pattern entries. On output, the number of entries written to *pattern*.

phase

On input, a pointer to a floating point value or `nil` if you do not want the phase. On output, this value contains the offset at which to start drawing the pattern, measured in points along the dashed-line pattern. For example, a phase of 6 in the pattern 5-2-3-2 would cause drawing to begin in the middle of the first gap.

Discussion

The array in the *pattern* parameter must be large enough to hold all of the returned values in the pattern. If you are not sure how many values there might be, you can call this method twice. The first time you call it, do not pass a value for *pattern* but use the returned value in *count* to allocate an array of floating-point numbers that you can then pass in the second time.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setLineDash:count:phase:` (page 330)

Declared In

`NSBezierPath.h`

isEmpty

Returns a Boolean value indicating whether the receiver is empty.

```
- (BOOL)isEmpty
```

Return Value

`YES` if the receiver contains no path elements; otherwise, `NO`.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Cropped Image

Declared In

`NSBezierPath.h`

lineCapStyle

Returns the line cap style for the receiver's path.

```
- (NSLineCapStyle)lineCapStyle
```

Return Value
The receiver's line cap style. For a list of values, see "Constants" (page 335). If this value is not set for the receiver, the default line cap style is returned.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ defaultLineCapStyle (page 298)
+ setDefaultLineCapStyle: (page 302)
- setLineCapStyle: (page 329)

Declared In
NSBezierPath.h

lineJoinStyle

Returns the receiver's line join style.

```
- (NSLineJoinStyle)lineJoinStyle
```

Return Value
The receiver's line join style. For a list of values, see "Constants" (page 335). If this value is not set for the receiver, the default line join style is returned.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ defaultLineJoinStyle (page 298)
+ setDefaultLineJoinStyle: (page 302)
- setLineJoinStyle: (page 330)

Declared In
NSBezierPath.h

lineToPoint:

Appends a straight line to the receiver's path

```
- (void)lineToPoint:(NSPoint)aPoint
```

Parameters
aPoint
The destination point of the line segment, specified in the current coordinate system.

Discussion
This method creates a straight line segment starting at the current point and ending at the point specified by the *aPoint* parameter. The current point is the last point in the receiver's most recently added segment.

You must set the path's current point (using the `moveToPoint:` (page 324) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `closePath` (page 315)
- `curveToPoint:controlPoint1:controlPoint2:` (page 317)

Related Sample Code
CompositeLab
FunHouse
NineSlice
Sketch+Accessibility
Sketch-112

Declared In
`NSBezierPath.h`

lineWidth

Returns the line width of the receiver's path.

```
- (CGFloat)lineWidth
```

Return Value
The line width of the receiver, measured in points in the user coordinate space.

Discussion
If no value was set explicitly for the receiver, this method returns the default line width.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setLineWidth:` (page 331)
+ `defaultLineWidth` (page 299)

Declared In
`NSBezierPath.h`

miterLimit

Returns the miter limit of the receiver's path.

```
- (CGFloat)miterLimit
```

Return Value
The miter limit of the path. If no value is set, this method returns the default miter limit.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMiterLimit:` (page 332)
+ `defaultMiterLimit` (page 299)

Declared In
`NSBezierPath.h`

moveToPoint:

Moves the receiver's current point to the specified location.

```
- (void)moveToPoint:(NSPoint)aPoint
```

Parameters

aPoint
A point in the current coordinate system.

Discussion
This method implicitly closes the current subpath (if any) and sets the current point to the value in *aPoint*. When closing the previous subpath, this method does not cause a line to be created from the first and last points in the subpath.

For many path operations, you must invoke this method before issuing any commands that cause a line or curve segment to be drawn.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `closePath` (page 315)
- `curveToPoint:controlPoint1:controlPoint2:` (page 317)
- `lineToPoint:` (page 322)

Related Sample Code
CompositeLab
NineSlice
Sketch-112
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSBezierPath.h`

relativeCurveToPoint:controlPoint1:controlPoint2:

Adds a Bezier cubic curve to the receiver's path from the current point to a new location, which is specified as a relative distance from the current point.

```
- (void)relativeCurveToPoint:(NSPoint)aPoint controlPoint1:(NSPoint)controlPoint1
    controlPoint2:(NSPoint)controlPoint2
```

Parameters

aPoint

The destination point of the curve segment, interpreted as a relative offset from the current point.

controlPoint1

The point that determines the shape of the curve near the current point, interpreted as a relative offset from the current point.

controlPoint2

The point that determines the shape of the curve near the destination point, interpreted as a relative offset from the current point.

Discussion

You must set the path's current point (using the `moveToPoint:` (page 324) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `closePath` (page 315)
- `curveToPoint:controlPoint1:controlPoint2:` (page 317)
- `relativeLineToPoint:` (page 325)
- `relativeMoveToPoint:` (page 326)

Declared In

`NSBezierPath.h`

relativeLineToPoint:

Appends a straight line segment to the receiver's path starting at the current point and moving towards the specified point, relative to the current location.

```
- (void)relativeLineToPoint:(NSPoint)aPoint
```

Parameters

aPoint

A point whose coordinates are interpreted as a relative offset from the current point.

Discussion

The destination point is relative to the current point. For example, if the current point is (1, 1) and *aPoint* contains the value (1, 2), a line segment is created between the points (1, 1) and (2, 3).

You must set the path's current point (using the `moveToPoint:` (page 324) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `closePath` (page 315)
- `lineToPoint:` (page 322)
- `relativeLineToPoint:` (page 325)
- `relativeMoveToPoint:` (page 326)

Related Sample Code
SampleRaster

Declared In
`NSBezierPath.h`

relativeMoveToPoint:

Moves the receiver's current point to a new point whose location is the specified distance from the current point.

```
- (void)relativeMoveToPoint:(NSPoint)aPoint
```

Parameters

aPoint

A point whose coordinates are interpreted as a relative offset from the current point.

Discussion
This method implicitly closes the current subpath (if any) and updates the location of the current point. For example, if the current point is (1, 1) and *aPoint* contains the value (1, 2), the previous subpath would be closed and the current point would become (2, 3). When closing the previous subpath, this method does not cause a line to be created from the first and last points in the subpath.

You must set the path's current point (using the `moveToPoint:` (page 324) method or through the creation of a preceding line or curve segment) before you invoke this method. If the path is empty, this method raises an `NSGenericException` exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `closePath` (page 315)
- `relativeCurveToPoint:controlPoint1:controlPoint2:` (page 325)
- `relativeLineToPoint:` (page 325)

Declared In
`NSBezierPath.h`

removeAllPoints

Removes all path elements from the receiver, effectively clearing the path.

```
- (void)removeAllPoints
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Cropped Image

Declared In
`NSBezierPath.h`

setAssociatedPoints:atIndex:

Changes the points associated with the specified path element.

```
- (void)setAssociatedPoints:(NSPointArray)points atIndex:(NSInteger)index
```

Parameters

points

A C-style array containing up to three `NSPoint` data types. This parameter must contain the correct number of points for the path element at the specified index. Move, close path, and line segment commands require one point. Curve operations require three points.

index

The index of the path element you want to modify.

Discussion
You can use this method to change the points associated with a path quickly and without recreating the path. You cannot use this method to change the type of the path element.

The following example shows you how you would modify the point associated with a line path element. The path created by this example results in a path with two elements. The first path element specifies a move to point (0, 0) while the second creates a line to point (100, 100). It then changes the line to go only to the point (50,50) using this method:

```
NSBezierPath *bezierPath = [NSBezierPath bezierPath];
NSPoint newPoint = NSMakePoint(50.0, 50.0);

[bezierPath moveToPoint: NSMakePoint(0.0, 0.0)];
[bezierPath lineToPoint: NSMakePoint(100.0, 100.0)];

// Modifies the point added by lineToPoint: method (100.0, 100.0)
// to the new point (50.0, 50.0)
[bezierPath setAssociatedPoints: &newPoint atIndex: 1];
```

Note: If you specify too few points for a path element of type `NSCurveToBezierPathElement`, the behavior of this method is undefined.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBezierPath.h`

setCachesBezierPath:

Sets whether the receiver should cache its path information. (Deprecated in Mac OS X v10.0.)

```
- (void)setCachesBezierPath:(BOOL)flag
```

Parameters

flag

YES if the receiver should cache its path information; otherwise, NO.

Discussion
Caching of paths currently has no effect.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.0.

See Also
- cachesBezierPath (page 315)

Declared In
NSBezierPath.h

setClip

Replaces the clipping path of the current graphics context with the area inside the receiver's path.

```
- (void)setClip
```

Discussion
You should avoid using this method as a way of adjusting the clipping path, as it may expand the clipping path beyond the bounds set by the enclosing view. If you do use this method, be sure to save the graphics state prior to modifying the clipping path and restore the graphics state when you are done.

This method uses the current winding rule to determine the clipping shape of the receiver. This method does not affect the receiver's path.

Availability
Available in Mac OS X v10.0 and later.

See Also
- addClip (page 306)
+ clipRect: (page 297)
- saveGraphicsState (page 1308) (NSGraphicsContext)
- restoreGraphicsState (page 1307) (NSGraphicsContext)

Related Sample Code
PDF Annotation Editor

Declared In
NSBezierPath.h

setFlatness:

Sets the flatness value for the receiver's path.

```
- (void)setFlatness:(CGFloat)flatness
```

Parameters

flatness

The flatness value for the path.

Discussion

The flatness value specifies the accuracy (or smoothness) with which curves are rendered. It is also the maximum error tolerance (measured in pixels) for rendering curves, where smaller numbers give smoother curves at the expense of more computation. The exact interpretation may vary slightly on different rendering devices.

The default flatness value is 0.6, which yields smooth curves.

Availability

Available in Mac OS X v10.0 and later.

See Also

- flatness (page 320)
+ setDefaultFlatness: (page 301)

Declared In

NSBezierPath.h

setLineCapStyle:

Sets the line cap style for the receiver's path.

```
- (void)setLineCapStyle:(NSLineCapStyle)lineCapStyle
```

Parameters

lineCapStyle

The line cap style to use with the receiver. For a list of values, see "Constants" (page 335).

Discussion

The line cap style specifies the shape of the endpoints of an open path when stroked. Figure 13-1 (page 302) shows the appearance of the available line cap styles.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ defaultLineCapStyle (page 298)
+ setDefaultLineCapStyle: (page 302)
- lineCapStyle (page 322)

Related Sample Code

DockTile
SpeedometerView
WebKitPluginStarter

WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

setLineDash:count:phase:

Sets the line-stroking pattern for the receiver.

```
- (void)setLineDash:(const CGFloat *)pattern count:(NSInteger)count
    phase:(CGFloat)phase
```

Parameters

pattern

A C-style array of floating point values that contains the lengths (measured in points) of the line segments and gaps in the pattern. The values in the array alternate, starting with the first line segment length, followed by the first gap length, followed by the second line segment length, and so on

count

The number of values in *pattern*.

phase

The offset at which to start drawing the pattern, measured in points along the dashed-line pattern. For example, a phase of 6 in the pattern 5-2-3-2 would cause drawing to begin in the middle of the first gap

Discussion
For example, to produce a supermarket coupon type of dashed line:

```
array[0] = 5.0; //segment painted with stroke color
array[1] = 2.0; //segment not painted with a color

[path setLineDash: array count: 2 phase: 0.0];
```

In the above example, if you set *phase* to 6.0, the line dash would begin exactly six units into *pattern*, which would start the pattern in the middle of the first gap.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `getLineDash:count:phase:` (page 321)

Related Sample Code
PhotoSearch

Declared In
NSBezierPath.h

setLineJoinStyle:

Sets the line join style for the receiver's path.

```
- (void)setLineJoinStyle:(NSLineJoinStyle)lineJoinStyle
```

Parameters

lineJoinStyle

The line join style to use for the receiver's path. For a list of values, see "Constants" (page 335).

Discussion

The line join style specifies the shape of the joints between connected segments of a stroked path. Figure 13-2 (page 303) shows the appearance of the available line join styles.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ defaultLineJoinStyle (page 298)
+ setDefaultLineJoinStyle: (page 302)
- lineJoinStyle (page 322)

Related Sample Code

DockTile
PDFKitLinker2
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In

NSBezierPath.h

setLineWidth:

Sets the line width of the receiver's path.

```
- (void)setLineWidth:(CGFloat)lineWidth
```

Parameters

lineWidth

The line width to use for the receiver, measured in points in the user coordinate space.

Discussion

The line width defines the thickness of the receiver's stroked path. A width of 0 is interpreted as the thinnest line that can be rendered on a particular device. The actual rendered line width may vary from the specified width by as much as 2 device pixels, depending on the position of the line with respect to the pixel grid and the current anti-aliasing settings. The width of the line may also be affected by scaling factors specified in the current transformation matrix of the active graphics context.

Availability

Available in Mac OS X v10.0 and later.

See Also

- lineWidth (page 323)
+ setDefaultLineWidth: (page 303)

Related Sample Code

DockTile
Sketch-112

SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

setMiterLimit:

Sets the miter limit for the receiver's path.

```
- (void)setMiterLimit:(CGFloat)miterLimit
```

Parameters
miterLimit
A value indicating the limit at which miter joins are converted to bevel joins.

Discussion
The miter limit helps you avoid spikes at the junction of two line segments connected by a miter join (NSMiterLineJoinStyle). If the ratio of the miter length—the diagonal length of the miter join—to the line thickness exceeds the miter limit, the joint is converted to a bevel join. The default miter limit value is 10, which converts miters whose angle at the joint is less than 11 degrees.

Availability
Available in Mac OS X v10.0 and later.

See Also
- miterLimit (page 323)
+ setDefaultMiterLimit: (page 304)

Declared In
NSBezierPath.h

setWindingRule:

Sets the winding rule used to fill the receiver's path.

```
- (void)setWindingRule:(NSWindingRule)aWindingRule
```

Parameters
aWindingRule
The winding rule to use for the path. This value may be either NSNonZeroWindingRule or NSEvenOddWindingRule.

Discussion
For more information on how winding rules affect the appearance of filled paths, see "Winding Rules and Filling Paths".

Availability
Available in Mac OS X v10.0 and later.

See Also
- fill (page 320)

- windingRule (page 334)
+ setDefaultWindingRule: (page 304)

Related Sample Code
Cropped Image

Declared In
NSBezierPath.h

stroke

Draws a line along the receiver's path using the current stroke color and drawing attributes.

```
- (void)stroke
```

Discussion
The drawn line is centered on the path with its sides parallel to the path segment. This method uses the current drawing attributes associated with the receiver. If a particular attribute is not set for the receiver, this method uses the corresponding default attribute.

Availability
Available in Mac OS X v10.0 and later.

See Also
- fill (page 320)
+ setDefaultLineCapStyle: (page 302)
+ setDefaultLineJoinStyle: (page 302)
- set (page 711) (NSColor)

Related Sample Code
DockTile
Sketch-112
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSBezierPath.h

transformUsingAffineTransform:

Transforms all points in the receiver using the specified transform.

```
- (void)transformUsingAffineTransform:(NSAffineTransform *)aTransform
```

Parameters
aTransform
The transform to apply to the path.

Discussion
This method applies the transform to the path's points immediately. The following code translates a line from 0,0 to 100,100 to a line from 10,10 to 110,110.

```
NSBezierPath *bezierPath = [NSBezierPath bezierPath];
NSAffineTransform *transform = [NSAffineTransform transform];

[bezierPath moveToPoint: NSMakePoint(0.0, 0.0)];
[bezierPath lineToPoint: NSMakePoint(100.0, 100.0)];

[transform translateXBy: 10.0 yBy: 10.0];
[bezierPath transformUsingAffineTransform: transform];
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CompositeLab
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSBezierPath.h`

windingRule

Returns the winding rule used to fill the receiver's path.

```
- (NSWindingRule)windingRule
```

Return Value
The winding rule for the path. This value may be either `NSNonZeroWindingRule` or `NSEvenOddWindingRule`.

Discussion
This value overrides the default value returned by `defaultWindingRule` (page 299).

For more information on how winding rules affect the appearance of filled paths, see "Winding Rules and Filling Paths".

Availability
Available in Mac OS X v10.0 and later.

See Also
- `fill` (page 320)
- `setWindingRule:` (page 332)
+ `defaultWindingRule` (page 299)

Declared In
`NSBezierPath.h`

Constants

NSBezierPathElement

Basic path element commands.

```
typedef enum {
    NSMoveToBezierPathElement,
    NSLineToBezierPathElement,
    NSCurveToBezierPathElement,
    NSClosePathBezierPathElement
} NSBezierPathElement;
```

Constants

`NSMoveToBezierPathElement`

Moves the path object's current drawing point to the specified point.

This path element does not result in any drawing. Using this command in the middle of a path results in a disconnected line segment.

Contains 1 point.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

`NSLineToBezierPathElement`

Creates a straight line from the current drawing point to the specified point.

Lines and rectangles are specified using this path element.

Contains 1 point.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

`NSCurveToBezierPathElement`

Creates a curved line segment from the current point to the specified endpoint using two control points to define the curve.

The points are stored in the following order: controlPoint1, controlPoint2, endPoint. Ovals, arcs, and Bezier curves all use curve elements to specify their geometry.

Contains 3 points.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

`NSClosePathBezierPathElement`

Marks the end of the current subpath at the specified point.

Note that the point specified for the Close Path element is essentially the same as the current point.

Available in Mac OS X v10.0 and later.

Declared in `NSBezierPath.h`.

Discussion

These commands are enough to define all of the possible path shapes. Each command has one or more points that contain information needed to position the path element. Most path elements use the current drawing point as the starting point for drawing. For more details, see Paths.

Declared In
NSBezierPath.h

NSLineJoinStyle

These constants specify the shape of the joints between connected segments of a stroked path.

```
typedef enum {
    NSMiterLineJoinStyle = 0,
    NSRoundLineJoinStyle = 1,
    NSBevelLineJoinStyle = 2
} NSLineJoinStyle;
```

Constants

`NSBevelLineJoinStyle`
Specifies a bevel line shape of the joints between connected segments of a stroked path.
See the `setDefaultLineJoinStyle:` (page 302) method for an example of the appearance.
Available in Mac OS X v10.0 and later.
Declared in `NSBezierPath.h`.

`NSMiterLineJoinStyle`
Specifies a miter line shape of the joints between connected segments of a stroked path.
See the `setDefaultLineJoinStyle:` (page 302) method for an example of the appearance.
Available in Mac OS X v10.0 and later.
Declared in `NSBezierPath.h`.

`NSRoundLineJoinStyle`
Specifies a round line shape of the joints between connected segments of a stroked path.
See the `setDefaultLineJoinStyle:` (page 302) method for an example of the appearance.
Available in Mac OS X v10.0 and later.
Declared in `NSBezierPath.h`.

Declared In
NSBezierPath.h

NSLineCapStyle

These constants specify the shape of endpoints for an open path when stroked.

```
typedef enum {
    NSButtLineCapStyle   = 0,
    NSRoundLineCapStyle  = 1,
    NSSquareLineCapStyle = 2
} NSLineCapStyle;
```

Constants

`NSButtLineCapStyle`
Specifies a butt line cap style for endpoints for an open path when stroked.
See the `setDefaultLineCapStyle:` (page 302) method for an example of the appearance.
Available in Mac OS X v10.0 and later.
Declared in `NSBezierPath.h`.

`NSSquareLineCapStyle`
: Specifies a square line cap style for endpoints for an open path when stroked.

  See the `setDefaultLineCapStyle:` (page 302) method for an example of the appearance.

  Available in Mac OS X v10.0 and later.

  Declared in `NSBezierPath.h`.

`NSRoundLineCapStyle`
: Specifies a round line cap style for endpoints for an open path when stroked.

  See the `setDefaultLineCapStyle:` (page 302) method for an example of the appearance.

  Available in Mac OS X v10.0 and later.

  Declared in `NSBezierPath.h`.

Declared In
`NSBezierPath.h`

NSWindingRule

These constants are used to specify the winding rule a Bezier path should use.

```
typedef enum {
    NSNonZeroWindingRule = 0,
    NSEvenOddWindingRule = 1
} NSWindingRule;
```

Constants
`NSNonZeroWindingRule`
: Specifies the non-zero winding rule.

  Count each left-to-right path as +1 and each right-to-left path as -1. If the sum of all crossings is 0, the point is outside the path. If the sum is nonzero, the point is inside the path and the region containing it is filled. This is the default winding rule.

  Available in Mac OS X v10.0 and later.

  Declared in `NSBezierPath.h`.

`NSEvenOddWindingRule`
: Specifies the even-odd winding rule.

  Count the total number of path crossings. If the number of crossings is even, the point is outside the path. If the number of crossings is odd, the point is inside the path and the region containing it should be filled.

  Available in Mac OS X v10.0 and later.

  Declared in `NSBezierPath.h`.

Discussion
These constants are described in more detail in Paths.

Declared In
`NSBezierPath.h`

NSBitmapImageRep Class Reference

Inherits from	NSImageRep : NSObject
Conforms to	NSCoding (NSImageRep) NSCopying (NSImageRep) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSBitmapImageRep.h
Companion guide	Cocoa Drawing Guide
Related sample code	GLSLShowpiece GLUT Image Difference Quartz EB Reducer

Overview

The `NSBitmapImageRep` class renders an image from bitmap data. Bitmap data formats supported include GIF, JPEG, TIFF, PNG, and various permutations of raw bitmap data.

Alpha Premultiplication

If a coverage (alpha) plane exists, a bitmap's color components are premultiplied with it. If you modify the contents of the bitmap, you are therefore responsible for premultiplying the data. For this reason, though, if you want to manipulate the actual data, an `NSBitmapImageRep` object is not recommended for storage. If you need to work with data that is not premultiplied, you should use Quartz, specifically `CGImageCreate` with `kCGImageAlphaLast`.

Note that premultiplying does not affect the output quality. Given source bitmap pixel `s`, destination pixel `d`, and alpha value `a`, a blend is basically

```
d' = a * s + (1 - a) * d
```

All premultiplication does is precalculate `a * s`.

Tasks

Creating an NSBitmapImageRep Object

`+ imageRepWithData:` (page 344)
Creates and returns an `NSBitmapImageRep` object initialized with the first image in the supplied data.

`+ imageRepsWithData:` (page 343)
Creates and returns an array of initialized `NSBitmapImageRep` objects corresponding to the images in the supplied data.

`- colorizeByMappingGray:toColor:blackMapping:whiteMapping:` (page 351)
Colorizes a grayscale image.

`- initWithBitmapDataPlanes:pixelsWide:pixelsHigh:bitsPerSample:samplesPerPixel:hasAlpha:isPlanar:colorSpaceName:bitmapFormat:bytesPerRow:bitsPerPixel:` (page 356)
Initializes the receiver, a newly allocated `NSBitmapImageRep` object, so it can render the specified image.

`- initWithBitmapDataPlanes:pixelsWide:pixelsHigh:bitsPerSample:samplesPerPixel:hasAlpha:isPlanar:colorSpaceName:bytesPerRow:bitsPerPixel:` (page 358)
Initializes the receiver, a newly allocated `NSBitmapImageRep` object, so it can render the specified image.

`- initWithCGImage:` (page 360)
Returns an `NSBitmapImageRep` object created from a Core Graphics image object.

`- initWithCIImage:` (page 361)
Returns an `NSBitmapImageRep` object created from a Core Image object.

`- initWithData:` (page 362)
Initializes a newly allocated `NSBitmapImageRep` from the provided data.

`- initWithFocusedViewRect:` (page 362)
Initializes the receiver, a newly allocated `NSBitmapImageRep` object, with bitmap data read from a rendered image.

`- initForIncrementalLoad` (page 355)
Initializes and returns the receiver, a newly allocated `NSBitmapImageRep` object, for incremental loading.

Getting Information About the Image

`- bitmapFormat` (page 347)
Returns the bitmap format of the receiver.

`- bitsPerPixel` (page 348)
Returns the number of bits allocated for each pixel in each plane of data.

`- bytesPerPlane` (page 349)
Returns the number of bytes in each plane or channel of data.

`- bytesPerRow` (page 349)
Returns the minimum number of bytes required to specify a scan line (a single row of pixels spanning the width of the image) in each data plane.

- `isPlanar` (page 363)
  Returns `YES` if image data is a planar configuration and `NO` if its in a meshed configuration.
- `numberOfPlanes` (page 363)
  Returns the number of separate planes image data is organized into.
- `samplesPerPixel` (page 364)
  Returns the number of components in the data.

Getting Image Data

- `bitmapData` (page 346)
  Returns a pointer to the bitmap data.
- `getBitmapDataPlanes:` (page 352)
  Returns by indirection bitmap data of the receiver separated into planes.

Producing Representations of the Image

+ `TIFFRepresentationOfImageRepsInArray:` (page 345)
  Returns a TIFF representation of the given images
+ `TIFFRepresentationOfImageRepsInArray:usingCompression:factor:` (page 346)
  Returns a TIFF representation of the given images using a specified compression scheme and factor.
- `TIFFRepresentation` (page 366)
  Returns a TIFF representation of the receiver.
- `TIFFRepresentationUsingCompression:factor:` (page 367)
  Returns a TIFF representation of the image using the specified compression.
+ `representationOfImageRepsInArray:usingType:properties:` (page 345)
  Formats the specified bitmap images using the specified storage type and properties and returns them in a data object.
- `representationUsingType:properties:` (page 363)
  Formats the receiver's image data using the specified storage type and properties and returns it in a data object.

Managing Compression Types

+ `getTIFFCompressionTypes:count:` (page 343)
  Returns by indirection an array of all available compression types that can be used when writing a TIFF image.
+ `localizedNameForTIFFCompressionType:` (page 344)
  Returns an autoreleased string containing the localized name for the specified compression type.
- `canBeCompressedUsing:` (page 350)
  Tests whether the receiver can be compressed by the specified compression scheme.
- `setCompression:factor:` (page 365)
  Sets the receiver's compression type and compression factor.

- `getCompression:factor:` (page 353)
  Returns by indirection the receiver's compression type and compression factor.
- `setProperty:withValue:` (page 366)
  Sets the image's *property* to *value*.
- `valueForProperty:` (page 368)
  Returns the value for the specified property.

Loading Image Incrementally

- `incrementalLoadFromData:complete:` (page 354)
  Loads the current image data into an incrementally-loaded image representation and returns the current status of the image.

Managing Pixel Values

- `setColor:atX:y:` (page 364)
  Changes the color of the pixel at the specified coordinates.
- `colorAtX:y:` (page 351)
  Returns the color of the pixel at the specified coordinates.
- `setPixel:atX:y:` (page 366)
  Sets the receiver's pixel at the specified coordinates to the specified raw pixel values.
- `getPixel:atX:y:` (page 354)
  Returns by indirection the pixel data for the specified location in the receiver.

Getting a Core Graphics Image

- `CGImage` (page 350)
  Returns a Core Graphics image object from the receiver's current bitmap data.

Managing ColorSpaces

- `bitmapImageRepByConvertingToColorSpace:renderingIntent:` (page 347)
  Converts the image rep to the specified colorspace
- `bitmapImageRepByRetaggingWithColorSpace:` (page 348)
  Changes the colorSpace tag of the receiver.
- `colorSpace` (page 352)
  Returns the image rep's colorSpace

Class Methods

getTIFFCompressionTypes:count:

Returns by indirection an array of all available compression types that can be used when writing a TIFF image.

```
+ (void)getTIFFCompressionTypes:(const NSTIFFCompression **)list count:(NSInteger
    *)numTypes
```

Parameters

list

On return, a C array of `NSTIFFCompression` constants. This array belongs to the `NSBitmapImageRep` class; it shouldn't be freed or altered. See "Constants" (page 368) for the supported TIFF compression types.

numTypes

The number of constants in list.

Discussion

Note that not all compression types can be used for all images: `NSTIFFCompressionNEXT` can be used only to retrieve image data. Because future releases may include other compression types, always use this method to get the available compression types—for example, when you implement a user interface for selecting compression types.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ localizedNameForTIFFCompressionType:` (page 344)
`- canBeCompressedUsing:` (page 350)

Declared In

`NSBitmapImageRep.h`

imageRepsWithData:

Creates and returns an array of initialized `NSBitmapImageRep` objects corresponding to the images in the supplied data.

```
+ (NSArray *)imageRepsWithData:(NSData *)bitmapData
```

Parameters

bitmapData

A data object containing one or more bitmapped images or `nil` if the class is unable to create an image representation. The *bitmapData* parameter can contain data in any supported bitmap format.

Return Value

An array of `NSBitmapImageRep` instances or an empty array if the class is unable to create any image representations.

Availability

Available in Mac OS X v10.0 and later.

Declared In
NSBitmapImageRep.h

imageRepWithData:

Creates and returns an NSBitmapImageRep object initialized with the first image in the supplied data.

```
+ (id)imageRepWithData:(NSData *)bitmapData
```

Parameters

bitmapData

A data object containing one or more bitmapped images. The *bitmapData* parameter can contain data in any supported bitmap format.

Return Value
An NSBitmapImageRep instance or nil if the class is unable to create an image representation.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
OpenCL Procedural Grass and Terrain Example
OpenCL_OceanWave
SpecialPictureProtocol

Declared In
NSBitmapImageRep.h

localizedNameForTIFFCompressionType:

Returns an autoreleased string containing the localized name for the specified compression type.

```
+ (NSString *)localizedNameForTIFFCompressionType:(NSTIFFCompression)compression
```

Parameters

compression

A TIFF compression type. NSTIFFCompression types are described in "Constants" (page 368).

Return Value
A localized name for *compression* or nil if *compression* is unrecognized.

Discussion
When implementing a user interface for selecting TIFF compression types, use getTIFFCompressionTypes:count: (page 343) to get the list of supported compression types, then use this method to get the localized names for each compression type.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ getTIFFCompressionTypes:count: (page 343)

Declared In
NSBitmapImageRep.h

representationOfImageRepsInArray:usingType:properties:

Formats the specified bitmap images using the specified storage type and properties and returns them in a data object.

```
+ (NSData *)representationOfImageRepsInArray:(NSArray *)imageReps
    usingType:(NSBitmapImageFileType)storageType properties:(NSDictionary
    *)properties
```

Parameters

imageReps

An array of NSBitmapImageRep objects.

storageType

An enum constant specifying a file type for bitmap images. It can be NSBMPFileType, NSGIFFileType, NSJPEGFileType, NSPNGFileType, or NSTIFFFileType.

properties

A dictionary that contains key-value pairs specifying image properties. These string constants used as keys and the valid values are described in "Bitmap image properties" (page 370).

Return Value
A data object containing the bitmap image data in the specified format. You can write this data to a file or use it to create a new NSBitmapImageRep object.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSBitmapImageRep.h

TIFFRepresentationOfImageRepsInArray:

Returns a TIFF representation of the given images

```
+ (NSData *)TIFFRepresentationOfImageRepsInArray:(NSArray *)array
```

Parameters

array

An array containing objects representing bitmap image representations.

Return Value
A data object containing a TIFF image representation.

Discussion
This method uses the compression returned by getCompression:factor: (page 353) (if applicable). If a problem is encountered during generation of the TIFF, this method raises an NSTIFFException or an NSBadBitmapParametersException.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `TIFFRepresentation` (page 366)

Declared In
`NSBitmapImageRep.h`

TIFFRepresentationOfImageRepsInArray:usingCompression:factor:

Returns a TIFF representation of the given images using a specified compression scheme and factor.

```
+ (NSData *)TIFFRepresentationOfImageRepsInArray:(NSArray *)array
    usingCompression:(NSTIFFCompression)compression factor:(float)factor
```

Parameters

array

An array containing objects representing bitmap image representations.

compression

An enum constant that represents a TIFF data-compression scheme. Legal values for *compression* can be found in `NSBitmapImageRep.h` and are described in "Constants" (page 368).

factor

A `float` value that provides a hint for those compression types that implement variable compression ratios.

Currently only JPEG compression uses a compression factor. JPEG compression in TIFF files is not supported, and *factor* is ignored.

Return Value
A data object containing a TIFF image representation.

Discussion
If the specified compression isn't applicable, no compression is used. If a problem is encountered during generation of the TIFF, the method raises an `NSTIFFException` or an `NSBadBitmapParametersException`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `TIFFRepresentationUsingCompression:factor:` (page 367)

Declared In
`NSBitmapImageRep.h`

Instance Methods

bitmapData

Returns a pointer to the bitmap data.

```
- (unsigned char *)bitmapData
```

Discussion
If the data is planar, returns a pointer to the first plane.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `getPixel:atX:y:` (page 354)
- `getBitmapDataPlanes:` (page 352)

Related Sample Code
Image Difference
LayerBackedOpenGLView
NSOpenGL Fullscreen
Quartz EB
SimpleImageFilter

Declared In
`NSBitmapImageRep.h`

bitmapFormat

Returns the bitmap format of the receiver.

```
- (NSBitmapFormat)bitmapFormat
```

Discussion
Returns 0 by default. The return value can indicate several different attributes, which are described in "Constants" (page 368).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `bytesPerRow` (page 349)

Declared In
`NSBitmapImageRep.h`

bitmapImageRepByConvertingToColorSpace:renderingIntent:

Converts the image rep to the specified colorspace

```
- (NSBitmapImageRep *)bitmapImageRepByConvertingToColorSpace:(NSColorSpace
    *)targetSpace renderingIntent:(NSColorRenderingIntent)renderingIntent
```

Parameters
targetSpace
The new colorSpace

renderingIntent
The rendering intent specifies how to handle colors that are not located within the target color space. The supported values are `NSColorRenderingIntent` (page 1313).

Return Value
An `NSBitmapImageRep`, or `nil`, if the conversion fails. If the original `NSBitmapImageRep` already uses that colorSpace, it is returned as is.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `bitmapImageRepByRetaggingWithColorSpace:` (page 348)
- `colorSpace` (page 352)

Declared In
`NSBitmapImageRep.h`

bitmapImageRepByRetaggingWithColorSpace:

Changes the colorSpace tag of the receiver.

```
- (NSBitmapImageRep *)bitmapImageRepByRetaggingWithColorSpace:(NSColorSpace
    *)newSpace
```

Parameters
newSpace
The desired colorSpace.

Return Value
An `NSBitmapImageRep`, or `nil`, if the conversion fails. If the original `NSBitmapImageRep` already uses that colorSpace, it is returned as is.

Discussion
This method will definitely fail if you pass a colorSpace that has a different color space model than the receiver. That is, if your original image is sRGB, you can only retag with some other RGB colorspace.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `bitmapImageRepByRetaggingWithColorSpace:` (page 348)
- `colorSpace` (page 352)

Declared In
`NSBitmapImageRep.h`

bitsPerPixel

Returns the number of bits allocated for each pixel in each plane of data.

```
- (NSInteger)bitsPerPixel
```

Discussion
This number is normally equal to the number of bits per sample or, if the data is in meshed configuration, the number of bits per sample times the number of samples per pixel. It can be explicitly set to another value (in `initWithBitmapDataPlanes:pixelsWide:pixelsHigh:bitsPerSample:samplesPerPixel:hasAlpha:isPlanar:colorSpaceName:bytesPerRow:bitsPerPixel:` (page 358)) in case extra memory is allocated for each pixel. This may be the case, for example, if pixel data is aligned on byte boundaries.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz EB

Declared In
`NSBitmapImageRep.h`

bytesPerPlane

Returns the number of bytes in each plane or channel of data.

```
- (NSInteger)bytesPerPlane
```

Discussion
This number is calculated from the number of bytes per row and the height of the image.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- bytesPerRow` (page 349)

Declared In
`NSBitmapImageRep.h`

bytesPerRow

Returns the minimum number of bytes required to specify a scan line (a single row of pixels spanning the width of the image) in each data plane.

```
- (NSInteger)bytesPerRow
```

Discussion
If not explicitly set to another value (in `initWithBitmapDataPlanes:pixelsWide:pixelsHigh:bitsPerSample:samplesPerPixel:hasAlpha:isPlanar:colorSpaceName:bytesPerRow:bitsPerPixel:` (page 358)), this number will be figured from the width of the image, the number of bits per sample, and, if the data is in a meshed configuration, the number of samples per pixel. It can be set to another value to indicate that each row of data is aligned on word or other boundaries.

Availability
Available in Mac OS X v10.0 and later.

See Also
- bytesPerPlane (page 349)

Related Sample Code
FunHouse
NURBSSurfaceVertexProg
OpenCL_OceanWave
Quartz EB
VertexPerformanceDemo

Declared In
NSBitmapImageRep.h

canBeCompressedUsing:

Tests whether the receiver can be compressed by the specified compression scheme.

```
- (BOOL)canBeCompressedUsing:(NSTIFFCompression)compression
```

Parameters
compression
A TIFF compression type. NSTIFFCompression types are defined in "Constants" (page 368).

Return Value
YES if the receiver's data matches compression with this type, NO if the data doesn't match *compression* or if *compression* is unsupported..

Discussion
Legal values for *compression* can be found in NSBitmapImageRep.h and are described in TIFF Compression in NSBitmapImageReps. This method returns YES if the receiver's data matches *compression*; for example, if *compression* is NSTIFFCompressionCCITTFAX3, then the data must be 1 bit per sample and 1 sample per pixel.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ getTIFFCompressionTypes:count: (page 343)

Declared In
NSBitmapImageRep.h

CGImage

Returns a Core Graphics image object from the receiver's current bitmap data.

```
- (CGImageRef)CGImage
```

Return Value
Returns an autoreleased CGImageRef opaque type based on the receiver's current bitmap data.

Discussion
The returned `CGImageRef` has pixel dimensions that are identical to the receiver's. This method might return a preexisting `CGImageRef` opaque type or create a new one. If the receiver is later modified, subsequent invocations of this method might return different `CGImageRef` opaque types.

Availability
Available in Mac OS X version 10.5.

See Also
`- initWithCGImage:` (page 360)

Declared In
`NSBitmapImageRep.h`

colorAtX:y:

Returns the color of the pixel at the specified coordinates.

```
- (NSColor *)colorAtX:(NSInteger)x y:(NSInteger)y
```

Parameters

x
The x-axis coordinate.

y
The y-axis coordinate.

Return Value
A color object representing the color at the specified coordinates.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setColor:atX:y:` (page 364)

Related Sample Code
RadiantColorPicker

Declared In
`NSBitmapImageRep.h`

colorizeByMappingGray:toColor:blackMapping:whiteMapping:

Colorizes a grayscale image.

```
- (void)colorizeByMappingGray:(CGFloat)midPoint toColor:(NSColor *)midPointColor
    blackMapping:(NSColor *)shadowColor whiteMapping:(NSColor *)lightColor
```

Parameters

midPoint
A float value representing the midpoint of the grayscale image.

midPointColor
A color object representing the midpoint of the color to map the image to.

shadowColor
A color object representing the black mapping to use for shadows.

lightColor
A color object representing the white mapping to be used in the image.

Discussion
This method maps the receiver such that:

Gray value of *midPoint* –> *midPointColor*;
black –> *shadowColor*;
white –> *lightColor*.

It works on images with 8-bit SPP, and thus supports either 8-bit gray or 24-bit color (with optional alpha).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBitmapImageRep.h`

colorSpace

Returns the image rep's colorSpace

```
- (NSColorSpace *)colorSpace
```

Return Value
The colorSpace.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- bitmapImageRepByRetaggingWithColorSpace:` (page 348)
`- bitmapImageRepByConvertingToColorSpace:renderingIntent:` (page 347)

Declared In
`NSBitmapImageRep.h`

getBitmapDataPlanes:

Returns by indirection bitmap data of the receiver separated into planes.

```
- (void)getBitmapDataPlanes:(unsigned char **)data
```

Parameters

data

On return, a C array of five character pointers. If the bitmap data is in planar configuration, each pointer will be initialized to point to one of the data planes. If there are less than five planes, the remaining pointers will be set to `NULL`. If the bitmap data is in meshed configuration, only the first pointer will be initialized; the others will be `NULL`.

Discussion

Color components in planar configuration are arranged in the expected order—for example, red before green before blue for RGB color. All color planes precede the coverage plane. If a coverage plane exists, the bitmap's color components are premultiplied with it. If you modify the contents of the bitmap, you are responsible for premultiplying the data.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- isPlanar` (page 363)

`- initWithBitmapDataPlanes:pixelsWide:pixelsHigh:bitsPerSample:samplesPerPixel:hasAlpha:isPlanar:colorSpaceName:bitmapFormat:bytesPerRow:bitsPerPixel:` (page 356)

`- initWithBitmapDataPlanes:pixelsWide:pixelsHigh:bitsPerSample:samplesPerPixel:hasAlpha:isPlanar:colorSpaceName:bytesPerRow:bitsPerPixel:` (page 358)

Declared In

`NSBitmapImageRep.h`

getCompression:factor:

Returns by indirection the receiver's compression type and compression factor.

```
- (void)getCompression:(NSTIFFCompression *)compression factor:(float *)factor
```

Parameters

compression

On return, an `enum` constant that represents the compression type used on the data; it corresponds to one of the values returned by the class method `getTIFFCompressionTypes:count:` (page 343).

factor

A float value that is specific to the compression type. Many types of compression don't support varying degrees of compression and thus ignore *factor*. JPEG compression allows a compression factor ranging from 0.0 to 1.0, with 0.0 being the lowest and 1.0 being the highest.

Discussion

Use this method to get information on the compression type for the source image data.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSBitmapImageRep.h`

getPixel:atX:y:

Returns by indirection the pixel data for the specified location in the receiver.

```
- (void)getPixel:(NSUInteger[])pixelData atX:(NSInteger)x y:(NSInteger)y
```

Parameters

pixelData

On return, an array of integers containing raw pixel data in the appropriate order for the receiver's `bitmapFormat` (page 347). Smaller integer samples, such as 4-bit, are returned as an integer. Floating point values are cast to integer values and returned.

x

The x-axis coordinate of the pixel.

y

The y-axis coordinate of the pixel.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setPixel:atX:y:` (page 366)

Declared In
`NSBitmapImageRep.h`

incrementalLoadFromData:complete:

Loads the current image data into an incrementally-loaded image representation and returns the current status of the image.

```
- (NSInteger)incrementalLoadFromData:(NSData *)data complete:(BOOL)complete
```

Parameters

data

A data object that contains the image to be loaded.

complete

`YES` if the image is entirely downloaded, `NO` otherwise.

Return Value
An integer constant indicating the status of the image during the load operation. See the discussion for details.

Discussion
After initializing the receiver with `initForIncrementalLoad` (page 355), you should call this method to incrementally load the image. Call this method each time new data becomes available. Always pass the entire image data buffer in *data*, not just the newest data, because the image decompressor may need the original data in order to backtrack. This method will block until the data is decompressed; it will decompress as much of the image as possible based on the length of the data. The image rep does not retain *data*, so you must ensure that *data* is not released for the duration of this method call. Pass `NO` for *complete* until the entire image is downloaded, at which time you should pass `YES`. You should also pass `YES` for *complete* if you have only partially downloaded the data, but cannot finish the download.

This method returns `NSImageRepLoadStatusUnknownType` if you did not pass enough data to determine the image format; you should continue to invoke this method with additional data.

This method returns `NSImageRepLoadStatusReadingHeader` if it has enough data to determine the image format, but needs more data to determine the size and depth and other characteristics of the image. You should continue to invoke this method with additional data.

This method returns `NSImageRepLoadStatusWillNeedAllData` if the image format does not support incremental loading or the Application Kit does not yet implement incremental loading for the image format. You may continue to invoke this method in this case, but until you pass `YES` for *`complete`*, this method will continue to return `NSImageRepLoadStatusWillNeedAllData`, and will perform no decompression. Once you pass `YES`, the image will be decompressed and one of the final three status messages will be returned.

If the image format does support incremental loading, then once enough data has been read, the image is decompressed from the top down a row at a time. In this case, instead of a status value, this method returns the number of pixel rows that have been decompressed, starting from the top of the image. You can use this information to draw the part of the image that is valid. The rest of the image is filled with opaque white. Note that if the image is progressive (as in a progressive JPEG or 2D interlaced PNG), this method may quickly return the full height of the image, but the image may still be loading, so do not use this return value as an indication of how much of the image remains to be decompressed.

If an error occurred while decompressing, this method returns `NSImageRepLoadStatusInvalidData`. If *`complete`* is `YES` but not enough data was available for decompression, `NSImageRepLoadStatusUnexpectedEOF` is returned. If enough data has been provided (regardless of the *`complete`* flag), then `NSImageRepLoadStatusCompleted` is returned. When any of these three status results are returned, this method has adjusted the `NSBitmapImageRep` so that `pixelsHigh` (page 1419) and `size` (page 1424), as well as the bitmap data, only contains the pixels that are valid, if any.

To cancel decompression, just pass in the existing data or `nil` and `YES` for *`complete`*. This method stops decompression immediately, adjusts the image size, and returns `NSImageRepLoadStatusUnexpectedEOF`. This method returns `NSImageRepLoadStatusCompleted` if you call it after receiving any error results (`NSImageRepLoadStatusInvalidData` or `NSImageRepLoadStatusUnexpectedEOF`) or if you call it on an `NSBitmapImageRep` that was not initialized with `initForIncrementalLoad` (page 355).

Availability
Available in Mac OS X v10.2 and later.

See Also
`- initForIncrementalLoad` (page 355)

Declared In
`NSBitmapImageRep.h`

initForIncrementalLoad

Initializes and returns the receiver, a newly allocated `NSBitmapImageRep` object, for incremental loading.

```
- (id)initForIncrementalLoad
```

Discussion
The receiver returns itself after setting its size and data buffer to zero. You can then call `incrementalLoadFromData:complete:` (page 354) to incrementally add image data.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- incrementalLoadFromData:complete:` (page 354)

Declared In
`NSBitmapImageRep.h`

initWithBitmapDataPlanes:pixelsWide:pixelsHigh:bitsPerSample:samplesPerPixel:hasAlpha:isPlanar:colorSpaceName:bitmapFormat:bytesPerRow:bitsPerPixel:

Initializes the receiver, a newly allocated `NSBitmapImageRep` object, so it can render the specified image.

```
- (id)initWithBitmapDataPlanes:(unsigned char **)planes pixelsWide:(NSInteger)width
    pixelsHigh:(NSInteger)height bitsPerSample:(NSInteger)bps
   samplesPerPixel:(NSInteger)spp hasAlpha:(BOOL)alpha isPlanar:(BOOL)isPlanar
   colorSpaceName:(NSString *)colorSpaceName
   bitmapFormat:(NSBitmapFormat)bitmapFormat bytesPerRow:(NSInteger)rowBytes
   bitsPerPixel:(NSInteger)pixelBits
```

Parameters

planes

An array of character pointers, each of which points to a buffer containing raw image data. If the data is in planar configuration, each buffer holds one component—one plane—of the data. Color planes are arranged in the standard order—for example, red before green before blue for RGB color. All color planes precede the coverage plane. If a coverage plane exists, the bitmap's color components must be premultiplied with it. If the data is in meshed configuration (that is, *isPlanar* is `NO`), only the first buffer is read.

If *planes* is `NULL` or an array of `NULL` pointers, this method allocates enough memory to hold the image described by the other arguments. You can then obtain pointers to this memory (with the `getPixel:atX:y:` (page 354) or `bitmapData` (page 346) method) and fill in the image data. In this case, the allocated memory will belong to the object and will be freed when it's freed.

If *planes* is not `NULL` and the array contains at least one data pointer, the returned object will only reference the image data; it will not copy it. The object treats the image data in the buffers as immutable and will not attempt to alter it. When the object itself is freed, it will not attempt to free the buffers.

width

The width of the image in pixels. This value must be greater than 0.

height

The height of the image in pixels. This value must be greater than 0.

bps

The number of bits used to specify one pixel in a single component of the data. All components are assumed to have the same bits per sample. *bps* should be one of these values: 1, 2, 4, 8, 12, or 16.

spp

The number of data components, or samples per pixel. This value includes both color components and the coverage component (alpha), if present. Meaningful values range from 1 through 5. An image with cyan, magenta, yellow, and black (CMYK) color components plus a coverage component would have an *spp* of 5; a grayscale image that lacks a coverage component would have an *spp* of 1.

alpha

YES if one of the components counted in the number of samples per pixel (*spp*) is a coverage (alpha) component, and NO if there is no coverage component. If YES, the color components in the bitmap data must be premultiplied with their coverage component.

isPlanar

YES if the data components are laid out in a series of separate "planes" or channels ("planar configuration") and NO if component values are interwoven in a single channel ("meshed configuration"). If NO, only the first buffer of planes is read.

For example, in meshed configuration, the red, green, blue, and coverage values for the first pixel of an image would precede the red, green, blue, and coverage values for the second pixel, and so on. In planar configuration, red values for all the pixels in the image would precede all green values, which would precede all blue values, which would precede all coverage values.

colorSpaceName

A string constant that indicates how data values are to be interpreted. It should be one of the following values:

- NSCalibratedWhiteColorSpace
- NSCalibratedBlackColorSpace
- NSCalibratedRGBColorSpace
- NSDeviceWhiteColorSpace
- NSDeviceBlackColorSpace
- NSDeviceRGBColorSpace
- NSDeviceCMYKColorSpace
- NSNamedColorSpace
- NSCustomColorSpace

If *bps* is 12, you cannot specify the monochrome color space.

bitmapFormat

An integer that specifies the ordering of the bitmap components. It is a mask created by combining the NSBitmapFormat constants NSAlphaFirstBitmapFormat, NSAlphaNonpremultipliedBitmapFormat and NSFloatingPointSamplesBitmapFormat using the C bitwise OR operator.

rowBytes

The number of bytes that are allocated for each scan line in each plane of data. A scan line is a single row of pixels spanning the width of the image.

Normally, *rowBytes* can be figured from the width of the image, the number of bits per pixel in each sample (*bps*), and, if the data is in a meshed configuration, the number of samples per pixel (*spp*). However, if the data for each row is aligned on word or other boundaries, it may have been necessary to allocate more memory for each row than there is data to fill it. *rowBytes* lets the object know whether that's the case.

If you pass in a *rowBytes* value of 0, the bitmap data allocated may be padded to fall on long word or larger boundaries for performance. If your code wants to advance row by row, use bytesPerRow (page 349) and do not assume the data is packed. Passing in a non-zero value allows you to specify exact row advances.

pixelBits

This integer value informs NSBitmapImageRep how many bits are actually allocated per pixel in each plane of data. If the data is in planar configuration, this normally equals *bps* (bits per sample). If the data is in meshed configuration, it normally equals *bps* times *spp* (samples per pixel). However, it's possible for a pixel specification to be followed by some meaningless bits (empty space), as may happen, for example, if pixel data is aligned on byte boundaries. NSBitmapImageRep supports only a limited number of *pixelBits* values (other than the default): for RGB images with 4 *bps*, *pixelBits* may be 16; for RGB images with 8 *bps*, *pixelBits* may be 32. The legal values for *pixelBits* are system dependent.

If you specify 0 for this parameter, the object interprets the number of bits per pixel using the values in the *bps* and *spp* parameters, as described in the preceding paragraph, without any meaningless bits.

Return Value
An initialized NSBitmapImageRep object or nil if the object cannot be initialized.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
Quartz Composer Offline Rendering

Declared In
NSBitmapImageRep.h

initWithBitmapDataPlanes:pixelsWide:pixelsHigh:bitsPerSample:samplesPerPixel:hasAlpha:isPlanar:colorSpaceName:bytesPerRow:bitsPerPixel:

Initializes the receiver, a newly allocated NSBitmapImageRep object, so it can render the specified image.

```
- (id)initWithBitmapDataPlanes:(unsigned char **)planes pixelsWide:(NSInteger)width
    pixelsHigh:(NSInteger)height bitsPerSample:(NSInteger)bps
   samplesPerPixel:(NSInteger)spp hasAlpha:(BOOL)alpha isPlanar:(BOOL)isPlanar
   colorSpaceName:(NSString *)colorSpaceName bytesPerRow:(NSInteger)rowBytes
   bitsPerPixel:(NSInteger)pixelBits
```

Parameters

planes

An array of character pointers, each of which points to a buffer containing raw image data. If the data is in planar configuration, each buffer holds one component—one plane—of the data. Color planes are arranged in the standard order—for example, red before green before blue for RGB color. All color planes precede the coverage plane. If a coverage plane exists, the bitmap's color components must be premultiplied with it. If the data is in meshed configuration (that is, *isPlanar* is NO), only the first buffer is read.

If *planes* is NULL or an array of NULL pointers, this method allocates enough memory to hold the image described by the other arguments. You can then obtain pointers to this memory (with the getPixel:atX:y: (page 354) or bitmapData (page 346) method) and fill in the image data. In this case, the allocated memory will belong to the object and will be freed when it's freed.

If *planes* is not NULL and the array contains at least one data pointer, the returned object will only reference the image data; it will not copy it. The object treats the image data in the buffers as immutable and will not attempt to alter it. When the object itself is freed, it will not attempt to free the buffers.

width

The width of the image in pixels. This value must be greater than 0.

height

The height of the image in pixels. This value must be greater than 0.

bps

The number of bits used to specify one pixel in a single component of the data. All components are assumed to have the same bits per sample. *bps* should be one of these values: 1, 2, 4, 8, 12, or 16.

spp

The number of data components, or samples per pixel. This value includes both color components and the coverage component (alpha), if present. Meaningful values range from 1 through 5. An image with cyan, magenta, yellow, and black (CMYK) color components plus a coverage component would have an *spp* of 5; a grayscale image that lacks a coverage component would have an *spp* of 1.

alpha

`YES` if one of the components counted in the number of samples per pixel (*spp*) is a coverage (alpha) component, and `NO` if there is no coverage component. If `YES`, the color components in the bitmap data must be premultiplied with their coverage component.

isPlanar

`YES` if the data components are laid out in a series of separate "planes" or channels ("planar configuration") and `NO` if component values are interwoven in a single channel ("meshed configuration"). If `NO`, only the first buffer of `planes` is read.

For example, in meshed configuration, the red, green, blue, and coverage values for the first pixel of an image would precede the red, green, blue, and coverage values for the second pixel, and so on. In planar configuration, red values for all the pixels in the image would precede all green values, which would precede all blue values, which would precede all coverage values.

colorSpaceName

A string constant that indicates how data values are to be interpreted. It should be one of the following values:

- `NSCalibratedWhiteColorSpace`
- `NSCalibratedBlackColorSpace`
- `NSCalibratedRGBColorSpace`
- `NSDeviceWhiteColorSpace`
- `NSDeviceBlackColorSpace`
- `NSDeviceRGBColorSpace`
- `NSDeviceCMYKColorSpace`
- `NSNamedColorSpace`
- `NSCustomColorSpace`

If *bps* is 12, you cannot specify the monochrome color space.

rowBytes

The number of bytes that are allocated for each scan line in each plane of data. A scan line is a single row of pixels spanning the width of the image.

Normally, *rowBytes* can be figured from the width of the image, the number of bits per pixel in each sample (*bps*), and, if the data is in a meshed configuration, the number of samples per pixel (*spp*). However, if the data for each row is aligned on word or other boundaries, it may have been necessary to allocate more memory for each row than there is data to fill it. *rowBytes* lets the object know whether that's the case.

If you pass in a *rowBytes* value of 0, the bitmap data allocated may be padded to fall on long word or larger boundaries for performance. If your code wants to advance row by row, use `bytesPerRow` (page 349) and do not assume the data is packed. Passing in a non-zero value allows you to specify exact row advances.

pixelBits

This integer value informs `NSBitmapImageRep` how many bits are actually allocated per pixel in each plane of data. If the data is in planar configuration, this normally equals *bps* (bits per sample). If the data is in meshed configuration, it normally equals *bps* times *spp* (samples per pixel). However, it's possible for a pixel specification to be followed by some meaningless bits (empty space), as may happen, for example, if pixel data is aligned on byte boundaries. `NSBitmapImageRep` supports only a limited number of *pixelBits* values (other than the default): for RGB images with 4 *bps*, *pixelBits* may be 16; for RGB images with 8 *bps*, *pixelBits* may be 32. The legal values for *pixelBits* are system dependent.

If you specify 0 for this parameter, the object interprets the number of bits per pixel using the values in the *bps* and *spp* parameters, as described in the preceding paragraph, without any meaningless bits.

Return Value
An initialized `NSBitmapImageRep` object or `nil` if the object cannot be initialized.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaVideoFrameToNSImage
ColorMatching
OpenCL NBody Simulation Example
Reducer
Transformed Image

Declared In
`NSBitmapImageRep.h`

initWithCGImage:

Returns an `NSBitmapImageRep` object created from a Core Graphics image object.

```
- (id)initWithCGImage:(CGImageRef)cgImage
```

Parameters

cgImage

A Core Graphics image object (an opaque type) from which to create the receiver. This opaque type is retained.

Return Value
An `NSBitmapImageRep` object initialized from the contents of the Core Graphics image or `nil` if the `NSBitmapImageRep` couldn't be created.

Discussion
If you use this method, you should treat the resulting bitmap `NSBitmapImageRep` object as read only. Because it only retains the value in the *`cgImage`* parameter, rather than unpacking the data, accessing the pixel data requires the creation of a copy of that data in memory. Changes to that data are not saved back to the Core Graphics image.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `CGImage` (page 350)

Declared In
`NSBitmapImageRep.h`

initWithCIImage:

Returns an `NSBitmapImageRep` object created from a Core Image object.

```
- (id)initWithCIImage:(CIImage *)ciImage
```

Parameters

`ciImage`

A Core Image object whose contents are to be copied to the receiver. This image rectangle must be of a finite size.

Return Value
An `NSBitmapImageRep` object initialized from the contents of the Core Image (`CIImage`) object or `nil` if the `NSBitmapImageRep` couldn't be created.

Discussion
The image in the *`ciImage`* parameter must be fully rendered before the receiver can be initialized. If you specify an object whose rendering was deferred (and thus does not have any pixels available now), this method forces the image to be rendered immediately. Rendering the image could result in a performance penalty if the image has a complex rendering chain or accelerated rendering hardware is not available. By the time this method returns, however, the resultant `NSBitmapImageRep` object can have its raw pixel data inspected, can be put on the pasteboard, and can be encoded to any of the standard image formats that `NSBitmapImageRep` supports (JPEG, TIFF, and so on.)

If you pass in a `CIImage` object whose extents are not finite, this method raises an exception.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `initWithBitmapImageRep:` (page 59) (CIImage)

Declared In
`NSBitmapImageRep.h`

initWithData:

Initializes a newly allocated `NSBitmapImageRep` from the provided data.

```
- (id)initWithData:(NSData *)bitmapData
```

Parameters

bitmapData

A data object containing image data. The contents of *bitmapData* can be any supported bitmap format. For TIFF data, the `NSBitmapImageRep` is initialized from the first header and image data found in *bitmapData*.

Return Value

Returns an initialized `NSBitmapImageRep` if the initialization was successful or `nil` if it was unable to interpret the contents of *bitmapData*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSBitmapImageRep.h`

initWithFocusedViewRect:

Initializes the receiver, a newly allocated `NSBitmapImageRep` object, with bitmap data read from a rendered image.

```
- (id)initWithFocusedViewRect:(NSRect)rect
```

Parameters

rect

A rectangle that specifies an area of the current window in the current coordinate system.

Return Value

Returns the initialized object or `nil` If for any reason the new object can't be initialized.

Discussion

This method uses imaging operators to read the image data into a buffer; the object is then created from that data. The object is initialized with information about the image obtained from the window server.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Cocoa OpenGL

Color Sampler

From A View to A Movie

FunHouse

Reducer

Declared In

`NSBitmapImageRep.h`

isPlanar

Returns YES if image data is a planar configuration and NO if its in a meshed configuration.

```
- (BOOL)isPlanar
```

Discussion
In a planar configuration, the image data is segregated into a separate plane for each color and coverage component. In a meshed configuration, the data is integrated into a single plane.

Availability
Available in Mac OS X v10.0 and later.

See Also
- samplesPerPixel (page 364)

Declared In
NSBitmapImageRep.h

numberOfPlanes

Returns the number of separate planes image data is organized into.

```
- (NSInteger)numberOfPlanes
```

Discussion
This number is the number of samples per pixel if the data has a separate plane for each component (isPlanar (page 363) returns YES) and 1 if the data is meshed (isPlanar (page 363) returns NO).

Availability
Available in Mac OS X v10.0 and later.

See Also
- samplesPerPixel (page 364)
- hasAlpha (page 1418) (NSImageRep)
- bitsPerSample (page 1414) (NSImageRep)

Declared In
NSBitmapImageRep.h

representationUsingType:properties:

Formats the receiver's image data using the specified storage type and properties and returns it in a data object.

```
- (NSData *)representationUsingType:(NSBitmapImageFileType)storageType
    properties:(NSDictionary *)properties
```

Parameters

storageType

An enum constant specifying a file type for bitmap images. It can be NSBMPFileType, NSGIFFileType, NSJPEGFileType, NSPNGFileType, or NSTIFFFileType.

properties

A dictionary that contains key-value pairs specifying image properties. These string constants used as keys and the valid values are described in "Bitmap image properties" (page 370).

Return Value
A data object containing the receiver's image data in the specified format. You can write this data to a file or use it to create a new `NSBitmapImageRep` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `TIFFRepresentation` (page 366)
- `TIFFRepresentationUsingCompression:factor:` (page 367)
- `TIFFRepresentation` (page 1372) (NSImage)
- `TIFFRepresentationUsingCompression:factor:` (page 1373) (NSImage)

Related Sample Code
Reducer
SpecialPictureProtocol

Declared In
`NSBitmapImageRep.h`

samplesPerPixel

Returns the number of components in the data.

```
- (NSInteger)samplesPerPixel
```

Discussion
The returned value includes both color components and the coverage component, if present.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hasAlpha` (page 1418) (NSImageRep)
- `bitsPerSample` (page 1414) (NSImageRep)

Related Sample Code
Image Difference

Declared In
`NSBitmapImageRep.h`

setColor:atX:y:

Changes the color of the pixel at the specified coordinates.

```
- (void)setColor:(NSColor *)color atX:(NSInteger)x y:(NSInteger)y
```

Parameters

color

A color object representing the color to be set.

x

The x-axis coordinate of the pixel.

y

The y-axis coordinate of the pixel.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `colorAtX:y:` (page 351)

Related Sample Code
RadiantColorPicker

Declared In
`NSBitmapImageRep.h`

setCompression:factor:

Sets the receiver's compression type and compression factor.

```
- (void)setCompression:(NSTIFFCompression)compression factor:(float)factor
```

Parameters

compression

An `enum` constant that identifies one of the supported compression types as described in "Constants" (page 368).

factor

A floating point value that is specific to the compression type. Many types of compression don't support varying degrees of compression and thus ignore *factor*. JPEG compression allows a compression factor ranging from 0.0 to 1.0, with 0.0 being the lowest and 1.0 being the highest.

Discussion
When an `NSBitmapImageRep` is created, the instance stores the compression type and factor for the source data. `TIFFRepresentation` (page 366) and `TIFFRepresentationOfImageRepsInArray:` (page 345) (class method) try to use the stored compression type and factor. Use this method to change the compression type and factor.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `canBeCompressedUsing:` (page 350)

Declared In
`NSBitmapImageRep.h`

setPixel:atX:y:

Sets the receiver's pixel at the specified coordinates to the specified raw pixel values.

```
- (void)setPixel:(NSUInteger[])pixelData atX:(NSInteger)x y:(NSInteger)y
```

Parameters

pixelData

An array of integers representing the raw pixel values. The values must be in an order appropriate to the receiver's `bitmapFormat` (page 347). Small pixel sample values should be passed as an integer value. Floating point values should be cast `int[]`.

x

The x-axis coordinate of the pixel.

y

The y-axis coordinate of the pixel.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- getPixel:atX:y:` (page 354)

Declared In
`NSBitmapImageRep.h`

setProperty:withValue:

Sets the image's *property* to *value*.

```
- (void)setProperty:(NSString *)property withValue:(id)value
```

Parameters

property

A string constant used as a key for an image property. These properties are described in "Constants" (page 368).

value

A value specific to `property`. If *value* is `nil`, the value of the property is cleared.

Discussion
The properties can affect how the image is read in and saved to file.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBitmapImageRep.h`

TIFFRepresentation

Returns a TIFF representation of the receiver.

```
- (NSData *)TIFFRepresentation
```

Discussion
This method invokes `TIFFRepresentationUsingCompression:factor:` (page 367) using the stored compression type and factor retrieved from the initial image data or changed using `setCompression:factor:` (page 365). If the stored compression type isn't supported for writing TIFF data (for example, `NSTIFFCompressionNEXT`), the stored compression is changed to `NSTIFFCompressionNone` before invoking `TIFFRepresentationUsingCompression:factor:` (page 367). receiver, using the compression that's returned by `getCompression:factor:` (page 353) (if applicable).

If a problem is encountered during generation of the TIFF, `TIFFRepresentation` raises an `NSTIFFException` or an `NSBadBitmapParametersException`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ TIFFRepresentationOfImageRepsInArray:` (page 345)
`- TIFFRepresentationUsingCompression:factor:` (page 367)
`- representationUsingType:properties:` (page 363)
`- TIFFRepresentation` (page 1372) (NSImage)
`- TIFFRepresentationUsingCompression:factor:` (page 1373) (NSImage)

Declared In
`NSBitmapImageRep.h`

TIFFRepresentationUsingCompression:factor:

Returns a TIFF representation of the image using the specified compression.

```
- (NSData *)TIFFRepresentationUsingCompression:(NSTIFFCompression)compression
    factor:(float)factor
```

Parameters

compression

An enum constant that represents a TIFF data-compression scheme. Legal values for *compression* can be found in `NSBitmapImageRep.h` and are described in "Constants" (page 368).

factor

A `float` value that provides a hint for those compression types that implement variable compression ratios.

Currently only JPEG compression uses a compression factor. JPEG compression in TIFF files is not supported, and *factor* is ignored.

Discussion
If the compression type isn't supported for writing TIFF data (for example, `NSTIFFCompressionNEXT`), the stored compression is changed to `NSTIFFCompressionNone` before the TIFF representation is generated.

If a problem is encountered during generation of the TIFF, `TIFFRepresentationUsingCompression:factor:` raises an `NSTIFFException` or an `NSBadBitmapParametersException`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- canBeCompressedUsing:` (page 350)
- `+ TIFFRepresentationOfImageRepsInArray:` (page 345)
- `- TIFFRepresentation` (page 366)
- `- representationUsingType:properties:` (page 363)
- `- TIFFRepresentation` (page 1372) (NSImage)
- `- TIFFRepresentationUsingCompression:factor:` (page 1373) (NSImage)

Related Sample Code
Quartz Composer Offline Rendering

Declared In
`NSBitmapImageRep.h`

valueForProperty:

Returns the value for the specified property.

```
- (id)valueForProperty:(NSString *)property
```

Parameters

property

A string constant used as a key for an image property. These properties are described in "Constants" (page 368).

Return Value
A value specific to `property`, or `nil` if the property is not set for the bitmap.

Discussion
Image properties can affect how an image is read in and saved to file. When retrieving the bitmap image properties defined in "Bitmap image properties" (page 370), be sure to check the return value of this method for a `nil` value. If a particular value is not set for the image, this method may return `nil`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBitmapImageRep.h`

Constants

NSImageRepLoadStatus

These constants represent the various status values returned by `incrementalLoadFromData:complete:` (page 354).

```
enum {
    NSImageRepLoadStatusUnknownType     = -1,
    NSImageRepLoadStatusReadingHeader   = -2,
    NSImageRepLoadStatusWillNeedAllData = -3,
    NSImageRepLoadStatusInvalidData     = -4,
    NSImageRepLoadStatusUnexpectedEOF   = -5,
    NSImageRepLoadStatusCompleted       = -6
};
typedef NSInteger NSImageRepLoadStatus;
```

Constants

`NSImageRepLoadStatusUnknownType`

Not enough data to determine image format. You should continue to provide more data.

Available in Mac OS X v10.2 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageRepLoadStatusReadingHeader`

The image format is known, but not enough data has been read to determine the size, depth, etc., of the image. You should continue to provide more data.

Available in Mac OS X v10.2 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageRepLoadStatusWillNeedAllData`

Incremental loading cannot be supported. Until you call `incrementalLoadFromData:complete:` (page 354) with `YES`, this status will be returned. You can continue to call the method but no decompression will take place. Once you do call the method with `YES`, then the image will be decompressed and one of the final three status messages will be returned.

Available in Mac OS X v10.2 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageRepLoadStatusInvalidData`

An error occurred during image decompression. The image contains the portions of the data that have already been successfully decompressed, if any

Available in Mac OS X v10.2 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageRepLoadStatusUnexpectedEOF`

`incrementalLoadFromData:complete:` (page 354) was called with `YES`, but not enough data was available for decompression. The image contains the portions of the data that have already been successfully decompressed, if any.

Available in Mac OS X v10.2 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageRepLoadStatusCompleted`

Enough data has been provided to successfully decompress the image (regardless of the complete: flag).

Available in Mac OS X v10.2 and later.

Declared in `NSBitmapImageRep.h`.

Bitmap image properties

These constants identify properties that are used by `representationOfImageRepsInArray:usingType:properties:` (page 345), `representationUsingType:properties:` (page 363), `setPixel:atX:y:` (page 366), and `valueForProperty:` (page 368).

```
NSString *NSImageCompressionMethod;
NSString *NSImageCompressionFactor;
NSString *NSImageDitherTransparency;
NSString *NSImageRGBColorTable;
NSString *NSImageInterlaced;
NSString *NSImageColorSyncProfileData;
NSString *NSImageFrameCount;
NSString *NSImageCurrentFrame;
NSString *NSImageCurrentFrameDuration;
NSString *NSImageLoopCount;
NSString *NSImageGamma;
NSString *NSImageProgressive;
NSString *NSImageEXIFData;
NSString *NSImageFallbackBackgroundColor;
```

Constants

`NSImageColorSyncProfileData`

Identifies an `NSData` object containing the ColorSync profile data.

It can be used for TIFF, JPEG, GIF, and PNG files. This value is set when reading in and used when writing out image data. You can get the profile data for a particular color space from the corresponding `NSColorSpace` object or from the ColorSync Manager.

Available in Mac OS X v10.0 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageCompressionFactor`

Identifies an `NSNumber` object containing the compression factor of the image.

Used only for JPEG files. JPEG compression in TIFF files is not supported, and the factor is ignored. The value is a float between 0.0 and 1.0, with 1.0 resulting in no compression and 0.0 resulting in the maximum compression possible. It's set when reading in and used when writing out the image.

Available in Mac OS X v10.0 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageCompressionMethod`

Identifies an `NSNumber` object identifying the compression method of the image.

Used only for TIFF files. The value corresponds to one of the `NSTIFFCompression` constants, described below. It's set when reading in and used when writing out.

Available in Mac OS X v10.0 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageDitherTransparency`

Identifies an `NSNumber` object containing a boolean that indicates whether the image is dithered.

Used only when writing GIF files.

Available in Mac OS X v10.0 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageInterlaced`

Identifies an `NSNumber` object containing a Boolean value that indicates whether the image is interlaced.

Used only when writing out PNG files.

Available in Mac OS X v10.0 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageRGBColorTable`

Identifies an `NSData` object containing the RGB color table.

Used only for GIF files. It's stored as packed RGB. It's set when reading in and used when writing out.

Available in Mac OS X v10.0 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageEXIFData`

Identifies an `NSDictionary` object containing the EXIF data for the image.

This property is used only when reading or writing JPEG files. The dictionary contains the EXIF keys and values. Th standard dictionary keys (that is, those that are not specific to camera vendors) are identical to those for `kCGImagePropertyExifDictionary` declared in the `CGImageSource` API. See `kCGImagePropertyExifDictionary` Keys for details.

Available in Mac OS X v10.4 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageFallbackBackgroundColor`

Specifies the background color to use when writing to an image format (such as JPEG) that doesn't support alpha. The color's alpha value is ignored. The default background color, when this property is not specified, is white. The value of the property should be an `NSColor` object. This constant corresponds to the `kCGImageDestinationBackgroundColor` constant in Quartz.

Available in Mac OS X v10.5 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageFrameCount`

Identifies an `NSNumber` object containing the number of frames in an animated GIF file.

This value is used when reading in data.

Available in Mac OS X v10.2 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageGamma`

Identifies an `NSNumber` object containing the gamma value for the image.

Used only for PNG files. The gamma values is a floating-point number between 0.0 and 1.0, with 0.0 being black and 1.0 being the maximum color. It's set when reading in and used when writing out.

Available in Mac OS X v10.4 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageCurrentFrame`

Identifies an `NSNumber` object containing the current frame for an animated GIF file.

The first frame is 0.

Available in Mac OS X v10.2 and later.

Declared in `NSBitmapImageRep.h`.

`NSImageCurrentFrameDuration`
: Identifies an `NSNumber` object containing the duration (in seconds) of the current frame for an animated GIF image.

  The frame duration can be a floating-point value. It is used when reading in, but not when writing out.

  Available in Mac OS X v10.2 and later.

  Declared in `NSBitmapImageRep.h`.

`NSImageProgressive`
: Identifies an `NSNumber` object containing a boolean that indicates whether the image uses progressive encoding.

  Used only for JPEG files. It's set when reading in and used when writing out.

  Available in Mac OS X v10.4 and later.

  Declared in `NSBitmapImageRep.h`.

`NSImageLoopCount`
: Identifies an `NSNumber` object containing the number of loops to make when animating a GIF image.

  A value of 0 indicates the animation should loop indefinitely. Values should be specified as integer numbers. It is used when reading in but not when writing out the image.

  Available in Mac OS X v10.3 and later.

  Declared in `NSBitmapImageRep.h`.

Discussion
When using the `valueForProperty:` method to retrieve the the value for any of these keys, be sure to check that the returned value is non-`nil` before you attempt to use it. A bitmap image representation may return `nil` for any values that have not yet been set.

NSBitmapImageFileType

The following file type constants are provided as a convenience by `NSBitmapImageRep`:

```
enum {
    NSTIFFFileType,
    NSBMPFileType,
    NSGIFFileType,
    NSJPEGFileType,
    NSPNGFileType,
    NSJPEG2000FileType
};
typedef NSUInteger NSBitmapImageFileType;
```

Constants
`NSTIFFFileType`
: Tagged Image File Format (TIFF)

  Available in Mac OS X v10.0 and later.

  Declared in `NSBitmapImageRep.h`.

`NSBMPFileType`
: Windows bitmap image (BMP) format

  Available in Mac OS X v10.0 and later.

  Declared in `NSBitmapImageRep.h`.

`NSGIFFileType`
Graphics Image Format (GIF), originally created by CompuServe for online downloads

Available in Mac OS X v10.0 and later.

Declared in `NSBitmapImageRep.h`.

`NSJPEGFileType`
JPEG format

Available in Mac OS X v10.0 and later.

Declared in `NSBitmapImageRep.h`.

`NSPNGFileType`
Portable Network Graphics (PNG) format

Available in Mac OS X v10.0 and later.

Declared in `NSBitmapImageRep.h`.

`NSJPEG2000FileType`
JPEG 2000 file format.

Available in Mac OS X v10.4 and later.

Declared in `NSBitmapImageRep.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBitmapImageRep.h`

NSTIFFCompression

These constants represent the various TIFF data-compression schemes supported by `NSBitmapImageRep`.

```
enum {
    NSTIFFCompressionNone = 1,
    NSTIFFCompressionCCITTFAX3 = 3,
    NSTIFFCompressionCCITTFAX4 = 4,
    NSTIFFCompressionLZW = 5,
    NSTIFFCompressionJPEG = 6,
    NSTIFFCompressionNEXT = 32766,
    NSTIFFCompressionPackBits = 32773,
    NSTIFFCompressionOldJPEG = 32865
};
typedef NSUInteger NSTIFFCompression;
```

Constants
`NSTIFFCompressionNone`
No compression.

Available in Mac OS X v10.0 and later.

Declared in `NSBitmapImageRep.h`.

`NSTIFFCompressionCCITTFAX3`
CCITT Fax Group 3 compression.

Used for 1-bit fax images sent over telephone lines.

Available in Mac OS X v10.0 and later.

Declared in `NSBitmapImageRep.h`.

`NSTIFFCompressionCCITTFAX4`
CCITT Fax Group 4 compression.
Used for 1-bit fax images sent over ISDN lines.
Available in Mac OS X v10.0 and later.
Declared in `NSBitmapImageRep.h`.

`NSTIFFCompressionLZW`
LZW compression.
Available in Mac OS X v10.0 and later.
Declared in `NSBitmapImageRep.h`.

`NSTIFFCompressionJPEG`
JPEG compression. No longer supported for input or output.
Available in Mac OS X v10.0 and later.
Declared in `NSBitmapImageRep.h`.

`NSTIFFCompressionNEXT`
NeXT compressed. Supported for input only.
Available in Mac OS X v10.0 and later.
Declared in `NSBitmapImageRep.h`.

`NSTIFFCompressionPackBits`
PackBits compression.
Available in Mac OS X v10.0 and later.
Declared in `NSBitmapImageRep.h`.

`NSTIFFCompressionOldJPEG`
Old JPEG compression. No longer supported for input or output.
Available in Mac OS X v10.0 and later.
Declared in `NSBitmapImageRep.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBitmapImageRep.h`

NSBitmapFormat

These constants represent the various bitmap component formats supported by `NSBitmapImageRep`. These values are combined using the C bitwise OR operator and passed to `initWithBitmapDataPlanes:pixelsWide:pixelsHigh:bitsPerSample:samplesPerPixel:hasAlpha:isPlanar:colorSpaceName:bitmapFormat:bytesPerRow:bitsPerPixel:` (page 356) as the bitmap format and are returned by `bitmapFormat` (page 347).

```
enum {
    NSAlphaFirstBitmapFormat            = 1 << 0,
    NSAlphaNonpremultipliedBitmapFormat = 1 << 1,
    NSFloatingPointSamplesBitmapFormat  = 1 << 2
};
typedef NSUInteger NSBitmapFormat;
```

Constants

`NSAlphaFirstBitmapFormat`

If 0, alpha values are the last component.

For example, CMYKA and RGBA.

Available in Mac OS X v10.4 and later.

Declared in `NSBitmapImageRep.h`.

`NSAlphaNonpremultipliedBitmapFormat`

If 0, alpha values are premultiplied.

Available in Mac OS X v10.4 and later.

Declared in `NSBitmapImageRep.h`.

`NSFloatingPointSamplesBitmapFormat`

If 0, samples are integer values.

Available in Mac OS X v10.4 and later.

Declared in `NSBitmapImageRep.h`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSBitmapImageRep.h`

NSBox Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSBox.h
Companion guide	Boxes
Related sample code	ButtonMadness CoreRecipes EnhancedDataBurn Quartz Composer QCTV TextSizingExample

Overview

The `NSBox` class implements simple views that can title themselves and draw a border around their content. These objects are known as **boxes**. You can use box to group, visually, some number of other views.

Subclassing Notes

An `NSBox` object is a view that draws a line around its rectangular bounds and that displays a title on or near the line (or might display neither line nor title). You can adjust the style of the line (bezel, grooved, or plain) as well as the placement and font of the title. An `NSBox` also has a content view to which other views can be added; it thus offers a way for an application to group related views. You could create a custom subclass of `NSBox` that alters or augments its appearance or that modifies its grouping behavior. For example, you might add color to the lines or background, add a new line style, or have the views in the group automatically snap to an invisible grid when added.

Methods to Override

You must override the `drawRect:` (page 3170) method (inherited from `NSView`) if you want to customize the appearance of your `NSBox` objects. Depending on the visual effect you're trying to achieve, you may have to invoke `super`'s implementation first. For example, if you are compositing a small image in a corner of the

box, you would invoke the superclass implementation first. If you're adding a new style of line, you would provide a way to store a request for this line type (such as a boolean instance variable and related accessor methods). Then, in `drawRect:` (page 3170), if a request for this line type exists, you would draw the entire view yourself (that is, without calling `super`). Otherwise, you would invoke the superclass implementation.

If you wish to change grouping behavior or other behavioral characteristics of the `NSBox` class, consider overriding `setContentView:` (page 385), `sizeToFit` (page 390), or `addSubview:` (page 3139) (inherited from `NSView`).

Special Considerations

If you are drawing the custom `NSBox` entirely by yourself, and you want it to look exactly like the superclass object (except for your changes), it may take some effort and time to get the details right.

Tasks

Configuring Boxes

- `borderRect` (page 380)
  Returns the rectangle in which the receiver's border is drawn.
- `boxType` (page 381)
  Returns the receiver's box type.
- `setBoxType:` (page 385)
  Sets the box type.
- `borderType` (page 380)
  Returns the receiver's border type.
- `setBorderType:` (page 384)
  Sets the border type to *`aType`*, which must be a valid border type.
- `isTransparent` (page 383)
  Indicates whether the receiver is transparent.
- `setTransparent:` (page 390)
  Specifies whether the receiver is transparent.
- `title` (page 391)
  Returns the receiver's title.
- `setTitle:` (page 388)
  Sets the title of the box and marks the region of the receiver within the title rectangle as needing display.
- `titleFont` (page 391)
  Returns the font object used to draw the receiver's title.
- `setTitleFont:` (page 388)
  Sets the font object used to draw the receiver's title and marks the region of the receiver within the title rectangle as needing display.
- `titlePosition` (page 392)
  Returns a constant representing the title position.

- `setTitlePosition:` (page 389)
  Sets the position of the box's title.
- `setTitleWithMnemonic:` (page 389)
  Sets the title of the receiver with a character denoted as an access key.
- `titleCell` (page 391)
  Returns the cell used to display the receiver's title.
- `titleRect` (page 392)
  Returns the rectangle in which the receiver's title is drawn.

Customizing

- `borderColor` (page 380)
  Returns the color of the receiver's border when the receiver is a custom box with a simple line border.
- `setBorderColor:` (page 384)
  Specifies the receiver's border color.
- `borderWidth` (page 381)
  Returns the width of the receiver's border when the receiver is a custom box with a simple line border.
- `setBorderWidth:` (page 385)
  Specifies the receiver's border width.
- `cornerRadius` (page 382)
  Returns the radius of the receiver's corners when the receiver is a custom box with a simple line border.
- `setCornerRadius:` (page 386)
  Specifies the receiver's corner radius.
- `fillColor` (page 383)
  Returns the color of the receiver's background when the receiver is a custom box with a simple line border.
- `setFillColor:` (page 387)
  Specifies the receiver's fill color.

Managing Content

- `contentView` (page 382)
  Returns the receiver's content view.
- `setContentView:` (page 385)
  Sets the receiver's content view.
- `contentViewMargins` (page 382)
  Returns the distances between the border and the content view.
- `setContentViewMargins:` (page 386)
  Sets the horizontal and vertical distance between the border of the receiver and its content view.

Sizing

- `setFrameFromContentFrame:` (page 387)
  Places the receiver so its content view lies on the specified frame.
- `sizeToFit` (page 390)
  Resizes and moves the receiver's content view so it just encloses its subviews.

Instance Methods

borderColor

Returns the color of the receiver's border when the receiver is a custom box with a simple line border.

```
- (NSColor *)borderColor
```

Return Value
The receiver's border color. It must be a custom box—that is, it has a type of `NSBoxCustom` (page 394)—and it must have a border style of `NSLineBorder` (page 3250).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setBorderColor:` (page 384)

Declared In
`NSBox.h`

borderRect

Returns the rectangle in which the receiver's border is drawn.

```
- (NSRect)borderRect
```

Return Value
The rectangle in which the border of the `NSBox` is drawn.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBox.h`

borderType

Returns the receiver's border type.

```
- (NSBorderType)borderType
```

Return Value
A constant describing the type of border. Border types are defined in `NSView.h`. Currently, the following border types are defined: `NSNoBorder`,`NSLineBorder`,`NSBezelBorder`, `NSGrooveBorder`.

By default, the border type of an `NSBox` is `NSGrooveBorder`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setBorderType:` (page 384)

Declared In
`NSBox.h`

borderWidth

Returns the width of the receiver's border when the receiver is a custom box with a simple line border.

```
- (CGFloat)borderWidth
```

Return Value
The receiver's border width. It must be a custom box—that is, it has a type of `NSBoxCustom` (page 394)—and it must have a border style of `NSLineBorder` (page 3250).

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setBorderWidth:` (page 385)

Declared In
`NSBox.h`

boxType

Returns the receiver's box type.

```
- (NSBoxType)boxType
```

Return Value
A constant describing the type of box. These constants are described in `NSBoxType` (page 394). By default, the box type of an `NSBox` is `NSBoxPrimary`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setBoxType:` (page 385)

Declared In
`NSBox.h`

contentView

Returns the receiver's content view.

```
- (id)contentView
```

Return Value
The content view of the `NSBox` object. The content view is created automatically when the box is created and resized as the box is resized (you should never send frame-altering messages directly to a box's content view). You can replace it with an `NSView` of your own through the `setContentView:` (page 385) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setContentView:` (page 385)

Related Sample Code
EnhancedDataBurn
Quartz Composer QCTV
TextSizingExample

Declared In
`NSBox.h`

contentViewMargins

Returns the distances between the border and the content view.

```
- (NSSize)contentViewMargins
```

Return Value
The width (the horizontal distance between the innermost edge of the border and the content view) and height (the vertical distance between the innermost edge of the border and the content view) describing the distance between the border and the content view. By default, these are both 5.0 in the box's coordinate system.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setContentViewMargins:` (page 386)

Declared In
`NSBox.h`

cornerRadius

Returns the radius of the receiver's corners when the receiver is a custom box with a simple line border.

```
- (CGFloat)cornerRadius
```

Return Value
The receiver's corner radius. It must be a custom box—that is, it has a type of `NSBoxCustom` (page 394)—and it must have a border style of `NSLineBorder` (page 3250).

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setCornerRadius:` (page 386)

Declared In
`NSBox.h`

fillColor

Returns the color of the receiver's background when the receiver is a custom box with a simple line border.

```
- (NSColor *)fillColor
```

Return Value
The receiver's fill color. It must be a custom box—that is, it has a type of `NSBoxCustom` (page 394)—and it must have a border style of `NSLineBorder` (page 3250).

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setFillColor:` (page 387)

Declared In
`NSBox.h`

isTransparent

Indicates whether the receiver is transparent.

```
- (BOOL)isTransparent
```

Return Value
`YES` when the receiver is transparent, `NO` otherwise.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setTransparent:` (page 390)

Declared In
`NSBox.h`

setBorderColor:

Specifies the receiver's border color.

```
- (void)setBorderColor:(NSColor *)borderColor
```

Parameters

borderColor

Border color for the receiver.

Special Considerations

Functional only when the receiver's box type (`boxType` (page 381)) is `NSBoxCustom` and its border type (`borderType` (page 380)) is `NSLineBorder`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `borderColor` (page 380)

Declared In
`NSBox.h`

setBorderType:

Sets the border type to *aType*, which must be a valid border type.

```
- (void)setBorderType:(NSBorderType)aType
```

Parameters

aType

A constant describing the type of border. Border types are defined in `NSView.h`. Currently, the following border types are defined: `NSNoBorder`,`NSLineBorder`,`NSBezelBorder`, `NSGrooveBorder`.

Discussion
If the size of the new border is different from that of the old border, the content view is resized to absorb the difference, and the box is marked for redisplay.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `borderType` (page 380)
- `setNeedsDisplay:` (page 3225) (`NSView`)

Related Sample Code
AnimatingViews
FunHouse

Declared In
`NSBox.h`

setBorderWidth:

Specifies the receiver's border width.

```
- (void)setBorderWidth:(CGFloat)borderWidth
```

Parameters

borderWidth

Border width for the receiver.

Special Considerations

Functional only when the receiver's box type (`boxType` (page 381)) is `NSBoxCustom` and its border type (`borderType` (page 380)) is `NSLineBorder`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `borderWidth` (page 381)

Declared In
`NSBox.h`

setBoxType:

Sets the box type.

```
- (void)setBoxType:(NSBoxType)boxType
```

Parameters

boxType

A constant describing the type of box; this must be a valid box type. These constants are described in `NSBoxType` (page 394).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `boxType` (page 381)

Related Sample Code
AnimatingViews
FunHouse

Declared In
`NSBox.h`

setContentView:

Sets the receiver's content view.

```
- (void)setContentView:(NSView *)aView
```

Parameters

aView

The new content view. The `NSView` object is resized to fit within the box's current content area and the box is marked for redisplay.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `contentView` (page 382)
- `setFrameFromContentFrame:` (page 387)
- `sizeToFit` (page 390)
- `setNeedsDisplay:` (page 3225) (`NSView`)

Related Sample Code
Quartz Composer QCTV

Declared In
`NSBox.h`

setContentViewMargins:

Sets the horizontal and vertical distance between the border of the receiver and its content view.

```
- (void)setContentViewMargins:(NSSize)offsetSize
```

Parameters

offsetSize

The width and height of the offset between the box's border and content view. The horizontal value is applied (reckoned in the box's coordinate system) fully and equally to the left and right sides of the box. The vertical value is similarly applied to the top and bottom.

Discussion
Unlike changing a box's other attributes, such as its title position or border type, changing the offsets doesn't automatically resize the content view. In general, you should send a `sizeToFit` (page 390) message to the box after changing the size of its offsets. This message causes the content view to remain unchanged while the box is sized to fit around it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `contentViewMargins` (page 382)

Declared In
`NSBox.h`

setCornerRadius:

Specifies the receiver's corner radius.

```
- (void)setCornerRadius:(CGFloat)cornerRadius
```

Parameters

cornerRadius

Corner radius for the receiver.

Special Considerations

Functional only when the receiver's box type (`boxType` (page 381)) is `NSBoxCustom` and its border type (`borderType` (page 380)) is `NSLineBorder`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- cornerRadius` (page 382)

Declared In
`NSBox.h`

setFillColor:

Specifies the receiver's fill color.

```
- (void)setFillColor:(NSColor *)fillColor
```

Parameters

fillColor

Fill color for the receiver.

Special Considerations

Functional only when the receiver's box type (`boxType` (page 381)) is `NSBoxCustom` and its border type (`borderType` (page 380)) is `NSLineBorder`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- fillColor` (page 383)

Related Sample Code
AnimatingViews

Declared In
`NSBox.h`

setFrameFromContentFrame:

Places the receiver so its content view lies on the specified frame.

```
- (void)setFrameFromContentFrame:(NSRect)contentFrame
```

Parameters

contentFrame

The rectangle specifying the frame of the box's content view, reckoned in the coordinate system of the box's superview. The box is marked for redisplay.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setContentViewMargins:` (page 386)
- `setFrame:` (page 3218) (`NSView`)
- `setNeedsDisplay:` (page 3225) (`NSView`)

Declared In
`NSBox.h`

setTitle:

Sets the title of the box and marks the region of the receiver within the title rectangle as needing display.

```
- (void)setTitle:(NSString *)aString
```

Parameters

aString

The new title of the `NSBox`. The default title of an `NSBox` is "Title." If the size of the new title is different from that of the old title, the content view is resized to absorb the difference.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `title` (page 391)
- `titleRect` (page 392)
- `setNeedsDisplayInRect:` (page 3225) (`NSView`)

Related Sample Code
Quartz Composer QCTV

Declared In
`NSBox.h`

setTitleFont:

Sets the font object used to draw the receiver's title and marks the region of the receiver within the title rectangle as needing display.

```
- (void)setTitleFont:(NSFont *)aFont
```

Parameters

aFont

The `NSFont` object used to draw the box's title.

Discussion
By default, the title is drawn using the small system font (obtained using (`smallSystemFontSize` (page 1157) as the parameter of `systemFontOfSize:` (page 1157), both `NSFont` class methods). If the size of the new font is different from that of the old font, the content view is resized to absorb the difference.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `titleFont` (page 391)
- `setNeedsDisplayInRect:` (page 3225) (`NSView`)

Declared In
`NSBox.h`

setTitlePosition:

Sets the position of the box's title.

```
- (void)setTitlePosition:(NSTitlePosition)aPosition
```

Parameters

aPosition

A constant describing the position of the box's title. These constants are described in `NSTitlePosition` (page 392). The default position is `NSAtTop`.

Discussion
If the new title position changes the size of the box's border area, the content view is resized to absorb the difference, and the box is marked as needing redisplay.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `titlePosition` (page 392)
- `setNeedsDisplay:` (page 3225) (`NSView`)

Related Sample Code
AnimatingViews
FunHouse

Declared In
`NSBox.h`

setTitleWithMnemonic:

Sets the title of the receiver with a character denoted as an access key.

```
- (void)setTitleWithMnemonic:(NSString *)aString
```

Discussion
Mnemonics are not supported in Mac OS X.

By default, a box's title is "Title." The content view is not automatically resized, and the box is not marked for redisplay.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTitleWithMnemonic:` (page 599) (`NSCell`)

Declared In
`NSBox.h`

setTransparent:

Specifies whether the receiver is transparent.

```
- (void)setTransparent:(BOOL)transparent
```

Parameters

transparent

`YES` makes the receiver transparent.

`NO` makes the receiver opaque.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `isTransparent` (page 383)

Declared In
`NSBox.h`

sizeToFit

Resizes and moves the receiver's content view so it just encloses its subviews.

```
- (void)sizeToFit
```

Discussion
The receiver is then moved and resized to wrap around the content view. The receiver's width is constrained so its title will be fully displayed.

You should invoke this method after:

- Adding a subview (to the content view)
- Altering the size or location of such a subview
- Setting the margins around the content view

The mechanism by which the content view is moved and resized depends on whether the object responds to its own `sizeToFit` message: If it does respond, then that message is sent, and the content view is expected to be so modified. If the content view doesn't respond, the box moves and resizes the content view itself.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBox.h`

title

Returns the receiver's title.

```
- (NSString *)title
```

Return Value
The title of the `NSBox`. By default, a box's title is "Title."

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitle:` (page 388)

Related Sample Code
Quartz Composer QCTV

Declared In
`NSBox.h`

titleCell

Returns the cell used to display the receiver's title.

```
- (id)titleCell
```

Return Value
The `NSCell` object used to display the title.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBox.h`

titleFont

Returns the font object used to draw the receiver's title.

```
- (NSFont *)titleFont
```

Return Value
The `NSFont` object used to draw the title.

Discussion
By default, the title is drawn using the small system font (obtained using (`smallSystemFontSize` (page 1157) as the parameter of `systemFontOfSize:` (page 1157), both `NSFont` class methods).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitleFont:` (page 388)

Declared In
NSBox.h

titlePosition

Returns a constant representing the title position.

```
- (NSTitlePosition)titlePosition
```

Return Value
A constant representing the position of the receiver's title. See `NSTitlePosition` (page 392) for a list of these constants.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTitlePosition:` (page 389)

Declared In
NSBox.h

titleRect

Returns the rectangle in which the receiver's title is drawn.

```
- (NSRect)titleRect
```

Return Value
The rectangle in which the title is drawn.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTitlePosition:` (page 389)
- `setTitle:` (page 388)
- `setTitleFont:` (page 388)
- `setFrameFromContentFrame:` (page 387)
- `sizeToFit` (page 390)

Declared In
NSBox.h

Constants

NSTitlePosition

Specify the location of a box's title with respect to its border.

```
typedef enum _NSTitlePosition {
    NSNoTitle     = 0,
    NSAboveTop    = 1,
    NSAtTop       = 2,
    NSBelowTop    = 3,
    NSAboveBottom = 4,
    NSAtBottom    = 5,
    NSBelowBottom = 6
} NSTitlePosition;
```

Constants

`NSNoTitle`
The box has no title.
Available in Mac OS X v10.0 and later.
Declared in `NSBox.h`.

`NSAboveTop`
Title positioned above the box's top border.
Available in Mac OS X v10.0 and later.
Declared in `NSBox.h`.

`NSAtTop`
Title positioned within the box's top border.
Available in Mac OS X v10.0 and later.
Declared in `NSBox.h`.

`NSBelowTop`
Title positioned below the box's top border.
Available in Mac OS X v10.0 and later.
Declared in `NSBox.h`.

`NSAboveBottom`
Title positioned above the box's bottom border.
Available in Mac OS X v10.0 and later.
Declared in `NSBox.h`.

`NSAtBottom`
Title positioned within the box's bottom border.
Available in Mac OS X v10.0 and later.
Declared in `NSBox.h`.

`NSBelowBottom`
Title positioned below the box's bottom border.
Available in Mac OS X v10.0 and later.
Declared in `NSBox.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBox.h`

NSBoxType

These constants and data type identifies box types, which, in conjunction with a box's border type, define the appearance of the box.

```
enum {
    NSBoxPrimary   = 0,
    NSBoxSecondary = 1,
    NSBoxSeparator = 2,
    NSBoxOldStyle  = 3,
    NSBoxCustom    = 4
};
typedef NSUInteger NSBoxType;
```

Constants

`NSBoxPrimary`

Specifies the primary box appearance. This is the default box type.

Available in Mac OS X v10.0 and later.

Declared in `NSBox.h`.

`NSBoxSecondary`

Specifies the secondary box appearance.

Available in Mac OS X v10.0 and later.

Declared in `NSBox.h`.

`NSBoxSeparator`

Specifies that the box is a separator.

Available in Mac OS X v10.0 and later.

Declared in `NSBox.h`.

`NSBoxOldStyle`

Specifies that the box is a Mac OS X v10.2–style box.

Available in Mac OS X v10.0 and later.

Declared in `NSBox.h`.

`NSBoxCustom`

Specifies that the appearance of the box is determined entirely by the by box-configuration methods, without automatically applying Apple human interface guidelines. See "`Customizing`" (page 379) for details.

Available in Mac OS X v10.5 and later.

Declared in `NSBox.h`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSBox.h`

NSBrowser Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSBrowser.h
Companion guide	Browsers
Related sample code	AnimatedTableView ComplexBrowser NewsReader SimpleCocoaBrowser ZipBrowser

Overview

This class provides a user interface for displaying and selecting items from a list of data or from hierarchically organized lists of data such as directory paths. Instances of this class are known as browsers. When working with a hierarchy of data, the levels are displayed in columns, which are indexed from left to right.

This class uses the `NSBrowserCell` class to implement its user interface.

Browsers have the following components:

- Columns
- Scroll views
- Matrices
- Browser cells

To the user, browsers display data in columns and rows within each column. These components are arranged in the following component hierarchy:

```
Browser
|---Columns [1..*]
    |---Scroll view
```

```
    |---Matrix
        |---Rows [0..*]
```

Tasks

Configuring Browsers

- `reusesColumns` (page 427)
  Indicates whether the browser reuses matrix objects after their columns are unloaded.
- `setReusesColumns:` (page 444)
  Specifies whether matrices can be reused.
- `maxVisibleColumns` (page 422)
  Returns the maximum number of visible columns.
- `setMaxVisibleColumns:` (page 442)
  Sets the maximum number of columns that can be displayed.
- `autohidesScroller` (page 406)
  Returns whether the browser automatically hides its scroller.
- `setAutohidesScroller:` (page 437)
  Allows the browser to hide its scroller automatically.
- `backgroundColor` (page 406)
  Provides the browser's background color.
- `setBackgroundColor:` (page 437)
  Specifies the browser's background color.
- `minColumnWidth` (page 423)
  Returns the minimum column width.
- `setMinColumnWidth:` (page 443)
  Sets the minimum column width.
- `separatesColumns` (page 435)
  Indicates whether columns are separated by bezeled borders.
- `setSeparatesColumns:` (page 446)
  Separates columns with bezeled borders.
- `takesTitleFromPreviousColumn` (page 448)
  Indicates whether a column takes its title from the selected cell in the previous column.
- `setTakesTitleFromPreviousColumn:` (page 447)
  Sets whether the title of a column is set to the string value of the selected cell in the previous column.
- `tile` (page 449)
  Adjusts the various subviews of the browser—scrollers, columns, titles, and so on—without redrawing.
- `delegate` (page 411)
  Returns the browser's delegate.
- `setDelegate:` (page 440)
  Sets the browser's delegate.

- `acceptsArrowKeys` (page 404) Deprecated in Mac OS X v10.6
  Indicates whether the browser allows navigation using the arrow keys. (Deprecated. There is no replacement.)
- `setAcceptsArrowKeys:` (page 435) Deprecated in Mac OS X v10.6
  Specifies whether the browser allows navigation using the arrow keys. (Deprecated. There is no replacement.)

Getting Browser Information

- `isOpaque` (page 418)
  Indicates whether the browser is opaque.

Managing Component Types

+ `cellClass` (page 403)
  Returns the `NSBrowserCell` class.
- `setCellClass:` (page 438)
  Sets the class of the cell to be used by the matrices in the columns of the browser.
- `cellPrototype` (page 407)
  Returns the browser's prototype `NSCell`.
- `setCellPrototype:` (page 438)
  Sets the prototype cell for displaying items in the matrices in the columns of the browser.
- `matrixClass` (page 421)
  Returns the matrix class used in the browser's columns.
- `setMatrixClass:` (page 442)
  Sets the matrix class to be used in the browser's columns.

Managing Selection Behavior

- `allowsBranchSelection` (page 404)
  Indicates whether the user can select branch items.
- `setAllowsBranchSelection:` (page 435)
  Allows the user to select branch items.
- `allowsEmptySelection` (page 405)
  Indicates whether there can be nothing selected.
- `setAllowsEmptySelection:` (page 436)
  Allows the user to select nothing.
- `allowsMultipleSelection` (page 405)
  Indicates whether the user can select multiple items.
- `setAllowsMultipleSelection:` (page 436)
  Allows the user to select multiple items.
- `selectedRowIndexesInColumn:` (page 432)
  Provides the indexes of the selected rows in a given column of the browser.

- `selectRowIndexes:inColumn:` (page 434)
  Specifies the selected rows in a given column of the browser.
- `allowsTypeSelect` (page 405)
  Indicates whether the browser allows keystroke-based selection (type select).
- `setAllowsTypeSelect:` (page 437)
  Allows the browser to accept keystroke-based selection.

Managing Selection

- `selectedCell` (page 430)
  Returns the last (rightmost and lowest) selected cell.
- `selectedCellInColumn:` (page 430)
  Returns the last (lowest) cell selected in the given column.
- `selectedCells` (page 431)
  Returns all cells selected in the rightmost column.
- `selectAll:` (page 430)
  Selects all cells in the last column of the browser.
- `selectedRowInColumn:` (page 432)
  Returns the row index of the selected cell in the specified column.
- `selectRow:inColumn:` (page 433)
  Selects the cell at the specified row and column index.
- `selectionIndexPath` (page 433)
  Returns the index path of the item selected in the browser.
- `setSelectionIndexPath:` (page 445)
  Sets the browser's selection to the item with the specified path.
- `selectionIndexPaths` (page 433)
  Returns an array containing the index paths of all items selected in the browser.
- `setSelectionIndexPaths:` (page 446)
  Sets the browser's selection to the items whose index paths are in the specified array.

Accessing Components

- `loadedCellAtRow:column:` (page 421)
  Loads, if necessary, and returns the cell at the specified row and column location.
- `matrixInColumn:` (page 422)
  Returns the matrix located in the specified column.
- `editItemAtIndexPath:withEvent:select:` (page 414)
  Begins editing the item at the specified path.
- `itemAtIndexPath:` (page 419)
  Returns the item at the specified index path.
- `itemAtRow:inColumn:` (page 420)
  Returns the item located at the specified row and column.

- `indexPathForColumn:` (page 417)
  Returns the index path of the item whose children are displayed in the given column.
- `isLeafItem:` (page 418)
  Returns whether the specified item is a leaf item.
- `parentForItemsInColumn:` (page 424)
  Returns the item that contains the children located in the specified column.

Managing the Path

- `path` (page 425)
  Returns a string representing the browser's current path.
- `setPath:` (page 443)
  Sets the path to be displayed by the browser.
- `pathToColumn:` (page 425)
  Returns a string representing the path from the first column up to, but not including, the column at the given index.
- `pathSeparator` (page 425)
  Returns the path separator.
- `setPathSeparator:` (page 444)
  Sets the path separator.

Managing Columns

- `addColumn` (page 404)
  Adds a column to the right of the last column.
- `columnOfMatrix:` (page 409)
  Returns the column number in which the given matrix is located.
- `selectedColumn` (page 431)
  Returns the index of the last column with a selected item.
- `lastColumn` (page 420)
  Returns the index of the last column loaded.
- `setLastColumn:` (page 442)
  Sets the last column.
- `firstVisibleColumn` (page 415)
  Returns the index of the first visible column.
- `numberOfVisibleColumns` (page 424)
  Returns the number of columns that are visible.
- `lastVisibleColumn` (page 420)
  Returns the index of the last visible column.
- `validateVisibleColumns` (page 451)
  Validates the browser's visible columns.
- `isLoaded` (page 418)
  Returns whether column 0 is loaded.

- `loadColumnZero` (page 421)
  Loads column 0; unloads previously loaded columns.
- `reloadColumn:` (page 426)
  Reloads the given column if it exists and sets it to be the last column.
- `displayAllColumns` (page 411) Deprecated in Mac OS X v10.3
  Updates the browser to display all loaded columns. (Deprecated. Use `setNeedsDisplayInRect:` (page 3225))
- `displayColumn:` (page 411) Deprecated in Mac OS X v10.3
  Updates the browser to display the given column. (Deprecated. Use `setNeedsDisplayInRect:` (page 3225) instead.)

Accessing Column Titles

- `titleOfColumn:` (page 450)
  Returns the title displayed for the given column.
- `setTitle:ofColumn:` (page 447)
  Sets the title of the given column.
- `isTitled` (page 419)
  Indicates whether columns display titles.
- `setTitled:` (page 447)
  Sets columns to display titles.
- `drawTitleOfColumn:inRect:` (page 414)
  Draws the title for the specified column within the given rectangle.
- `titleHeight` (page 450)
  Returns the height of the column titles.
- `titleFrameOfColumn:` (page 449)
  Returns the bounds of the title frame for the specified column.

Updating Browsers

- `noteHeightOfRowsWithIndexesChanged:inColumn:` (page 423)
  Immediately retiles the browser's columns using row heights specified by the browser's delegate.
- `reloadDataForRowIndexes:inColumn:` (page 427)
  Updates the rows in the column with the specified column index with indexes in the specified set.

Scrolling

- `hasHorizontalScroller` (page 417)
  Indicates whether the browser has a horizontal scroller.
- `setHasHorizontalScroller:` (page 441)
  Sets whether the browser has a scroller to scroll horizontally.
- `scrollColumnToVisible:` (page 428)
  Scrolls to make the specified column visible.

- `scrollColumnsLeftBy:` (page 428)
  Scrolls columns left by the specified number of columns.
- `scrollColumnsRightBy:` (page 428)
  Scrolls columns right by the specified number of columns.
- `scrollRowToVisible:inColumn:` (page 429)
  Scrolls the specified row to be visible within the specified column.
- `scrollViaScroller:` (page 429) Deprecated in Mac OS X v10.3
  Scrolls columns left or right based on an `NSScroller`. (Deprecated. There is no replacement.)
- `updateScroller` (page 450) Deprecated in Mac OS X v10.3
  Updates the horizontal scroller to reflect column positions. (Deprecated. There is no replacement.)

Dragging

- `draggingSourceOperationMaskForLocal:` (page 413)
  Indicates the types of dragging operations the source object allows on the dragged image's data.
- `setDraggingSourceOperationMask:forLocal:` (page 441)
  Specifies the drag-operation mask for dragging operations with local or external destinations.
- `canDragRowsWithIndexes:inColumn:withEvent:` (page 407)
  Indicates whether the browser can attempt to initiate a drag of the given rows for the given event.
- `draggingImageForRowsWithIndexes:inColumn:withEvent:offset:` (page 413)
  Provides an image to represent dragged rows during a drag operation on the browser.
- `namesOfPromisedFilesDroppedAtDestination:` (page 423)
  Provides the names of the files that the browser promises to create at a specified location.

Getting Column Frames

- `frameOfColumn:` (page 415)
  Returns the rectangle containing the given column.
- `frameOfInsideOfColumn:` (page 415)
  Returns the rectangle containing the specified column, not including borders.

Getting Row Frames

- `frameOfRow:inColumn:` (page 416)
  Returns the frame of the cell at the specified location, including the expandable arrow.
- `getRow:column:forPoint:` (page 416)
  Gets the row and column coordinates for the specified point, if a cell exists at that point.

Managing Actions

- `doubleAction` (page 412)
  Returns the browser's double-click action method.

- `setDoubleAction:` (page 440)
  Sets the browser's double-click action.
- `sendsActionOnArrowKeys` (page 434)
  Returns whether pressing an arrow key causes an action message to be sent.
- `setSendsActionOnArrowKeys:` (page 446)
  Allows the specified action message to be sent when the user presses an arrow key.
- `sendAction` (page 434)
  Sends the action message to the target.

Handling Mouse-Click Events

- `doClick:` (page 412)
  Responds to (single) mouse clicks in a column of the browser.
- `doDoubleClick:` (page 412)
  Responds to double clicks in a column of the browser.
- `clickedColumn` (page 407)
  Returns the column number of the cell that the user clicked to display a context menu.
- `clickedRow` (page 408)
  Returns the row number of the cell that the user clicked to display a context menu.

Sizing

+ `removeSavedColumnsWithAutosaveName:` (page 403)
  Removes the column configuration data stored under the given name from the application's user defaults.
- `columnsAutosaveName` (page 409)
  Returns the name used to automatically save the browser's column configuration.
- `setColumnsAutosaveName:` (page 439)
  Sets the name used to automatically save the browser's column configuration.
- `columnContentWidthForColumnWidth:` (page 408)
  Returns the content width for a given column width.
- `columnWidthForColumnContentWidth:` (page 410)
  Returns the column width for the width of the given column's content.
- `columnResizingType` (page 409)
  Returns the browser's column resizing type.
- `setColumnResizingType:` (page 438)
  Sets the browser's column resizing type.
- `prefersAllColumnUserResizing` (page 426)
  Returns whether the browser is set to resize all columns simultaneously rather than resizing a single column at a time.
- `setPrefersAllColumnUserResizing:` (page 444)
  Causes the browser to resize all column simultaneously rather than resize a single column at a time.
- `widthOfColumn:` (page 451)
  Returns the width of the specified column.

- `setWidth:ofColumn:` (page 448)
  Sets the width of the specified column.
- `defaultColumnWidth` (page 410)
  Returns the default column width of the browser's columns.
- `setDefaultColumnWidth:` (page 439)
  Sets the default column width for new browser columns that do not otherwise have an initial width from defaults or the browser's delegate.
- `rowHeight` (page 427)
  Returns the height of the browser's rows.
- `setRowHeight:` (page 445)
  Sets the height of the browser's rows to the specified value.

Class Methods

cellClass

Returns the `NSBrowserCell` class.

```
+ (Class)cellClass
```

Return Value
Always returns the `NSBrowserCell` class (even if the developer has sent a `setCellClass:` (page 438) message to a particular instance).

Discussion
This method is used by `NSControl` during initialization and is not meant to be used by applications.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cellPrototype` (page 407)
- `setCellPrototype:` (page 438)

Declared In
`NSBrowser.h`

removeSavedColumnsWithAutosaveName:

Removes the column configuration data stored under the given name from the application's user defaults.

```
+ (void)removeSavedColumnsWithAutosaveName:(NSString *)name
```

Parameters
name
The name of the column configuration data to remove.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setColumnsAutosaveName:` (page 439)

Declared In
`NSBrowser.h`

Instance Methods

acceptsArrowKeys

Indicates whether the browser allows navigation using the arrow keys. (Deprecated in Mac OS X v10.6. There is no replacement.)

```
- (BOOL)acceptsArrowKeys
```

Return Value
`YES` if the arrow keys are enabled; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `setAcceptsArrowKeys:` (page 435)

Declared In
`NSBrowser.h`

addColumn

Adds a column to the right of the last column.

```
- (void)addColumn
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedColumn` (page 431)

Declared In
`NSBrowser.h`

allowsBranchSelection

Indicates whether the user can select branch items.

```
- (BOOL)allowsBranchSelection
```

Return Value
YES if the user can select branch items when multiple selection is enabled; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setAllowsBranchSelection: (page 435)

Declared In
NSBrowser.h

allowsEmptySelection

Indicates whether there can be nothing selected.

```
- (BOOL)allowsEmptySelection
```

Return Value
YES if the browser allows the selection to be empty; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setAllowsEmptySelection: (page 436)

Declared In
NSBrowser.h

allowsMultipleSelection

Indicates whether the user can select multiple items.

```
- (BOOL)allowsMultipleSelection
```

Return Value
YES if the browser allows the user to select multiple items at once; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setAllowsMultipleSelection: (page 436)

Declared In
NSBrowser.h

allowsTypeSelect

Indicates whether the browser allows keystroke-based selection (type select).

```
- (BOOL)allowsTypeSelect
```

Return Value
`YES` (default) when the browser allows keystroke-based selection; otherwise, `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setAllowsTypeSelect:` (page 437)

Declared In
`NSBrowser.h`

autohidesScroller

Returns whether the browser automatically hides its scroller.

```
- (BOOL)autohidesScroller
```

Return Value
`YES` if the scroller is automatically hidden; otherwise, `NO`.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setAutohidesScroller:` (page 437)

Declared In
`NSBrowser.h`

backgroundColor

Provides the browser's background color.

```
- (NSColor *)backgroundColor
```

Return Value
The browser's background color.

Discussion
The default value is `[NSColor whiteColor]`

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setBackgroundColor:` (page 437)

Declared In
`NSBrowser.h`

canDragRowsWithIndexes:inColumn:withEvent:

Indicates whether the browser can attempt to initiate a drag of the given rows for the given event.

```
- (BOOL)canDragRowsWithIndexes:(NSIndexSet *)rowIndexes
    inColumn:(NSInteger)columnIndex withEvent:(NSEvent *)dragEvent
```

Parameters

rowIndexes

Rows the user is dragging

columnIndex

Column containing the rows the user is dragging.

dragEvent

Mouse-drag event.

Return Value

`YES` when *rowIndexes* identifies at least one row and all the identified rows are enabled; otherwise, `NO`.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `browser:canDragRowsWithIndexes:inColumn:withEvent:` (page 3585) (`NSBrowserDelegate`)

Declared In

`NSBrowser.h`

cellPrototype

Returns the browser's prototype `NSCell`.

```
- (id)cellPrototype
```

Return Value

The prototype `NSCell`. The prototype `NSCell` instance is copied to display items in the matrices of the browser.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setCellPrototype:` (page 438)
- `setCellClass:` (page 438)

Declared In

`NSBrowser.h`

clickedColumn

Returns the column number of the cell that the user clicked to display a context menu.

```
- (NSInteger)clickedColumn
```

Return Value
The column number of the clicked cell, or `-1` if no context menu is active.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- clickedRow` (page 408)

Declared In
`NSBrowser.h`

clickedRow

Returns the row number of the cell that the user clicked to display a context menu.

```
- (NSInteger)clickedRow
```

Return Value
The row number of the clicked cell, or `-1` if no context menu is active.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- clickedColumn` (page 407)

Declared In
`NSBrowser.h`

columnContentWidthForColumnWidth:

Returns the content width for a given column width.

```
- (CGFloat)columnContentWidthForColumnWidth:(CGFloat)columnWidth
```

Parameters
columnWidth
The width of the column. This width is the entire scrolling text view.

Return Value
The width of the content for the column. This is the width of the matrix in the column.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- columnWidthForColumnContentWidth:` (page 410)

Declared In
`NSBrowser.h`

columnOfMatrix:

Returns the column number in which the given matrix is located.

```
- (NSInteger)columnOfMatrix:(NSMatrix *)matrix
```

Parameters

matrix

The matrix for which to return the column number.

Return Value

The index of the column in which the specified matrix appears.

Availability

Available in Mac OS X v10.0 and later.

See Also

- matrixInColumn: (page 422)

Declared In

NSBrowser.h

columnResizingType

Returns the browser's column resizing type.

```
- (NSBrowserColumnResizingType)columnResizingType
```

Return Value

A constant indicating the column resizing type. Possible return values are described in "NSBrowserColumnResizingType" (page 451). The default is NSBrowserAutoColumnResizing (page 452)

Availability

Available in Mac OS X v10.3 and later.

See Also

- setColumnResizingType: (page 438)

Declared In

NSBrowser.h

columnsAutosaveName

Returns the name used to automatically save the browser's column configuration.

```
- (NSString *)columnsAutosaveName
```

Return Value

The name used to save the column configuration.

Availability

Available in Mac OS X v10.3 and later.

See Also
- setColumnsAutosaveName: (page 439)

Declared In
NSBrowser.h

columnWidthForColumnContentWidth:

Returns the column width for the width of the given column's content.

```
- (CGFloat)columnWidthForColumnContentWidth:(CGFloat)columnContentWidth
```

Parameters

columnContentWidth
The width of the column's content (the width of the the matrix in the column).

Return Value
The width of the column (the width of the entire scrolling text view).

Discussion
For example, to guarantee that 16 pixels of your browser cell are always visible, call:

```
[browser setMinColumnWidth: [browser columnWidthForColumnContentWidth:16]]
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- columnContentWidthForColumnWidth: (page 408)

Declared In
NSBrowser.h

defaultColumnWidth

Returns the default column width of the browser's columns.

```
- (CGFloat)defaultColumnWidth
```

Return Value
The default column width.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setDefaultColumnWidth: (page 439)

Declared In
NSBrowser.h

delegate

Returns the browser's delegate.

```
- (id <NSBrowserDelegate>)delegate
```

Return Value
The browser's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setDelegate: (page 440)

Declared In
NSBrowser.h

displayAllColumns

Updates the browser to display all loaded columns. (Deprecated in Mac OS X v10.3. Use setNeedsDisplayInRect: (page 3225))

```
- (void)displayAllColumns
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

See Also
- addColumn (page 404)
- validateVisibleColumns (page 451)

Declared In
NSBrowser.h

displayColumn:

Updates the browser to display the given column. (Deprecated in Mac OS X v10.3. Use setNeedsDisplayInRect: (page 3225) instead.)

```
- (void)displayColumn:(NSInteger)column
```

Parameters
column
The index of the column to display.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

See Also
- addColumn (page 404)

- validateVisibleColumns (page 451)

Declared In
NSBrowser.h

doClick:

Responds to (single) mouse clicks in a column of the browser.

```
- (void)doClick:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- sendAction (page 434)

Declared In
NSBrowser.h

doDoubleClick:

Responds to double clicks in a column of the browser.

```
- (void)doDoubleClick:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- setDoubleAction: (page 440)

Declared In
NSBrowser.h

doubleAction

Returns the browser's double-click action method.

```
- (SEL)doubleAction
```

Return Value
The action method invoked when the user double-clicks on the browser.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setDoubleAction: (page 440)

Declared In
NSBrowser.h

draggingImageForRowsWithIndexes:inColumn:withEvent:offset:

Provides an image to represent dragged rows during a drag operation on the browser.

```
- (NSImage *)draggingImageForRowsWithIndexes:(NSIndexSet *)rowIndexes
    inColumn:(NSInteger)columnIndex withEvent:(NSEvent *)dragEvent
    offset:(NSPointPointer)dragImageOffset
```

Parameters

rowIndexes

Rows the user is dragging.

columnIndex

Column with the rows the user is dragging.

dragEvent

Mouse drag event.

dragImageOffset

Offset for the returned image:

- `NSZeroPoint`: The image is centered under the pointer.

Return Value

Image representing the visible cells identified by rowIndexes.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- browser:draggingImageForRowsWithIndexes:inColumn:withEvent:offset:` (page 3587) (`NSBrowserDelegate`)

Related Sample Code

ComplexBrowser

ZipBrowser

Declared In

`NSBrowser.h`

draggingSourceOperationMaskForLocal:

Indicates the types of dragging operations the source object allows on the dragged image's data.

```
- (NSDragOperation)draggingSourceOperationMaskForLocal:(BOOL)localDestination
```

Parameters

localDestination

Indicates the location of the dragging operation's destination object: `YES` for this application, `NO` for another application.

Return Value

- `NSDragOperationEvery` (page 3665) when localDestination is `YES`.
- `NSDragOperationNone` (page 3666) when localDestination is `NO`.

Discussion
This method overrides `NSDraggingSource draggingSourceOperationMaskForLocal:` (page 3670).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setDraggingSourceOperationMask:forLocal:` (page 441)

Declared In
`NSDragging.h`

drawTitleOfColumn:inRect:

Draws the title for the specified column within the given rectangle.

```
- (void)drawTitleOfColumn:(NSInteger)column inRect:(NSRect)aRect
```

Parameters

column
The index of the column for which to draw the title.

aRect
The rectangle within which to draw the title.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitle:ofColumn:` (page 447)
`- titleFrameOfColumn:` (page 449)
`- titleHeight` (page 450)

Declared In
`NSBrowser.h`

editItemAtIndexPath:withEvent:select:

Begins editing the item at the specified path.

```
- (void)editItemAtIndexPath:(NSIndexPath *)indexPath
    withEvent:(NSEvent *)theEvent
    select:(BOOL)select
```

Parameters

indexPath
The path of the item.

theEvent
The event to use when beginning the edit.

select
If `YES`, the cells contents will be selected; if `NO`, they will not be selected.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSBrowser.h

firstVisibleColumn

Returns the index of the first visible column.

```
- (NSInteger)firstVisibleColumn
```

Return Value
The index of the first visible column.

Availability
Available in Mac OS X v10.0 and later.

See Also
- lastVisibleColumn (page 420)

Declared In
NSBrowser.h

frameOfColumn:

Returns the rectangle containing the given column.

```
- (NSRect)frameOfColumn:(NSInteger)column
```

Parameters
column
The index of the column for which to retrieve the frame.

Return Value
The rectangle containing the specified column.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSBrowser.h

frameOfInsideOfColumn:

Returns the rectangle containing the specified column, not including borders.

```
- (NSRect)frameOfInsideOfColumn:(NSInteger)column
```

Parameters
column
The index of the column for which to retrieve the inside frame.

Return Value
The rectangle containing the column, not including the column borders.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSBrowser.h

frameOfRow:inColumn:

Returns the frame of the cell at the specified location, including the expandable arrow.

```
- (NSRect)frameOfRow:(NSInteger)row
    inColumn:(NSInteger)column
```

Parameters

row
The row of the cell.

column
The column of the cell.

Return Value
The frame of the cell, in the `NSBrowser` coordinate space.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSBrowser.h

getRow:column:forPoint:

Gets the row and column coordinates for the specified point, if a cell exists at that point.

```
- (BOOL)getRow:(NSInteger *)row
    column:(NSInteger *)column
    forPoint:(NSPoint)point
```

Parameters

row
On output, the row number of the cell at the specified point, or `-1` if there is no cell at the point.

column
On output, he column number of the cell at the specified point, or `-1` if there is no cell at the point.

point
The point to test.

Return Value
`YES` if a cell exists at the specified point; otherwise, `NO`.

Discussion
If a row does not exist at *point*, then `-1` is set for the row. If a column does not exist at *point*, then `-1` is set for the column.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSBrowser.h

hasHorizontalScroller

Indicates whether the browser has a horizontal scroller.

```
- (BOOL)hasHorizontalScroller
```

Return Value
YES if the browser uses an NSScroller object to scroll horizontally; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setHasHorizontalScroller: (page 441)

Declared In
NSBrowser.h

indexPathForColumn:

Returns the index path of the item whose children are displayed in the given column.

```
- (NSIndexPath *)indexPathForColumn:(NSInteger)column
```

Parameters

column
The column to find the index path for.

Return Value
The index path of the column.

Discussion
This method can only be used if the delegate implements the item data source methods.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
ComplexBrowser
ZipBrowser

Declared In
NSBrowser.h

isLeafItem:

Returns whether the specified item is a leaf item.

```
- (BOOL)isLeafItem:(id)item
```

Parameters

item

The item to be checked.

Return Value

`YES` if the item is a leaf item; otherwise, `NO`.

Discussion

This method may return `NO` if the item has never been displayed in the browser or accessed via `itemAtIndexPath:` (page 419). Overriding this method has no effect. It may be used only if the browser's delegate implements the item data source methods.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSBrowser.h`

isLoaded

Returns whether column 0 is loaded.

```
- (BOOL)isLoaded
```

Return Value

`YES` if column 0 is loaded; otherwise, `NO`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `loadColumnZero` (page 421)
- `reloadColumn:` (page 426)

Declared In

`NSBrowser.h`

isOpaque

Indicates whether the browser is opaque.

```
- (BOOL)isOpaque
```

Return Value

`YES` when the browser doesn't have a title and its background color's alpha component is `1.0`; otherwise, `NO`.

Discussion
This method overrides the `isOpaque` (page 3183) method of `NSView`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

isTitled

Indicates whether columns display titles.

```
- (BOOL)isTitled
```

Return Value
`YES` if the columns in a browser display titles; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitled:` (page 447)

Declared In
`NSBrowser.h`

itemAtIndexPath:

Returns the item at the specified index path.

```
- (id)itemAtIndexPath:(NSIndexPath *)indexPath
```

Parameters
indexPath
The index path of the item to return.

Return Value
The item.

Discussion
This method can only be used if the delegate implements the item data source methods. The specified index path must be displayable in the browser.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
ComplexBrowser
ZipBrowser

Declared In
`NSBrowser.h`

itemAtRow:inColumn:

Returns the item located at the specified row and column.

```
- (id)itemAtRow:(NSInteger)row
    inColumn:(NSInteger)column
```

Parameters

row

The row of the item.

column

The column of the item.

Return Value
The item.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSBrowser.h

lastColumn

Returns the index of the last column loaded.

```
- (NSInteger)lastColumn
```

Return Value
The index of the last loaded column.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setLastColumn: (page 442)

Declared In
NSBrowser.h

lastVisibleColumn

Returns the index of the last visible column.

```
- (NSInteger)lastVisibleColumn
```

Return Value
The index of the last visible column.

Availability
Available in Mac OS X v10.0 and later.

See Also
- firstVisibleColumn (page 415)

Declared In
NSBrowser.h

loadColumnZero

Loads column 0; unloads previously loaded columns.

```
- (void)loadColumnZero
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- isLoaded (page 418)
- reloadColumn: (page 426)

Related Sample Code
ZipBrowser

Declared In
NSBrowser.h

loadedCellAtRow:column:

Loads, if necessary, and returns the cell at the specified row and column location.

```
- (id)loadedCellAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters
row
The row index of the cell to return.
column
The column index of the cell to return.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectedCellInColumn: (page 430)
- selectRow:inColumn: (page 433)

Declared In
NSBrowser.h

matrixClass

Returns the matrix class used in the browser's columns.

```
- (Class)matrixClass
```

Return Value
The class of `NSMatrix` used in the browser's columns.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMatrixClass:` (page 442)

Declared In
`NSBrowser.h`

matrixInColumn:

Returns the matrix located in the specified column.

```
- (NSMatrix *)matrixInColumn:(NSInteger)column
```

Parameters
column
The column index of the matrix to obtain.

Return Value
The matrix located in the column.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- columnOfMatrix:` (page 409)

Declared In
`NSBrowser.h`

maxVisibleColumns

Returns the maximum number of visible columns.

```
- (NSInteger)maxVisibleColumns
```

Return Value
The maximum number of visible columns.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMaxVisibleColumns:` (page 442)

Declared In
`NSBrowser.h`

minColumnWidth

Returns the minimum column width.

```
- (CGFloat)minColumnWidth
```

Return Value
The minimum column width, in pixels.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setMinColumnWidth: (page 443)

Declared In
NSBrowser.h

namesOfPromisedFilesDroppedAtDestination:

Provides the names of the files that the browser promises to create at a specified location.

```
- (NSArray *)namesOfPromisedFilesDroppedAtDestination:(NSURL *)dropDestination
```

Return Value
Result of sending
browser:namesOfPromisedFilesDroppedAtDestination:forDraggedRowsWithIndexes:inColumn:
to the delegate.

Discussion
Implementation of NSDraggingSource namesOfPromisedFilesDroppedAtDestination: (page 3671).

Availability
Available in Mac OS X v10.2 and later.

See Also
- browser:namesOfPromisedFilesDroppedAtDestination:forDraggedRowsWithIndexes:inColumn: (page 3589) (NSBrowserDelegate)

Declared In
NSDragging.h

noteHeightOfRowsWithIndexesChanged:inColumn:

Immediately retiles the browser's columns using row heights specified by the browser's delegate.

```
- (void)noteHeightOfRowsWithIndexesChanged:(NSIndexSet *)indexSet
    inColumn:(NSInteger)columnIndex
```

Parameters
indexSet
The indexes of the rows to resize.

columnIndex
The column to retile.

Discussion
The browser's delegate must implement

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSBrowser.h

numberOfVisibleColumns

Returns the number of columns that are visible.

```
- (NSInteger)numberOfVisibleColumns
```

Return Value
The number of visible columns.

Availability
Available in Mac OS X v10.0 and later.

See Also
- validateVisibleColumns (page 451)

Declared In
NSBrowser.h

parentForItemsInColumn:

Returns the item that contains the children located in the specified column.

```
- (id)parentForItemsInColumn:(NSInteger)column
```

Parameters
column
The column.

Return Value
The parent item for the specified column.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
AnimatedTableView
ComplexBrowser

Declared In
NSBrowser.h

path

Returns a string representing the browser's current path.

```
- (NSString *)path
```

Return Value
The path representing the current selection. The components of this path are separated with the string returned by `pathSeparator` (page 425).

Discussion
Invoking this method is equivalent to invoking `pathToColumn:` (page 425) for all columns.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPath:` (page 443)
- `pathToColumn:` (page 425)

Declared In
`NSBrowser.h`

pathSeparator

Returns the path separator.

```
- (NSString *)pathSeparator
```

Return Value
The path separator. The default is "/".

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPathSeparator:` (page 444)

Declared In
`NSBrowser.h`

pathToColumn:

Returns a string representing the path from the first column up to, but not including, the column at the given index.

```
- (NSString *)pathToColumn:(NSInteger)column
```

Parameters
column
The index of the column at which the path stops.

Return Value
The path of the current selection up to, but not including, the specified column. The components of this path are separated with the string returned by `pathSeparator` (page 425).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `path` (page 425)
- `setPath:` (page 443)

Declared In
`NSBrowser.h`

prefersAllColumnUserResizing

Returns whether the browser is set to resize all columns simultaneously rather than resizing a single column at a time.

```
- (BOOL)prefersAllColumnUserResizing
```

Return Value
`YES` if the browser is set to resize all columns simultaneously; otherwise, `NO`. The default is `NO`.

Discussion
This setting applies only to browsers that allow the user to resize columns (see the constant `NSBrowserUserColumnResizing` (page 452). Holding down the Option key while resizing switches the type of resizing used.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setPrefersAllColumnUserResizing:` (page 444)

Declared In
`NSBrowser.h`

reloadColumn:

Reloads the given column if it exists and sets it to be the last column.

```
- (void)reloadColumn:(NSInteger)column
```

Parameters
column
The index of the column to reload.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isLoaded` (page 418)

- loadColumnZero (page 421)

Related Sample Code
NewsReader
ZipBrowser

Declared In
NSBrowser.h

reloadDataForRowIndexes:inColumn:

Updates the rows in the column with the specified column index with indexes in the specified set.

```
- (void)reloadDataForRowIndexes:(NSIndexSet *)rowIndexes
    inColumn:(NSInteger)column
```

Parameters

rowIndexes
The set of row indexes of the rows to be updated.

column
The column containing the rows to be updated.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSBrowser.h

reusesColumns

Indicates whether the browser reuses matrix objects after their columns are unloaded.

```
- (BOOL)reusesColumns
```

Return Value
YES if NSMatrix objects aren't freed when their columns are unloaded; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setReusesColumns: (page 444)

Declared In
NSBrowser.h

rowHeight

Returns the height of the browser's rows.

```
- (CGFloat)rowHeight
```

Return Value
The row height.

Discussion
The default value is 17.0. `rowHeight` and `setRowHeight:` (page 445) are only available when using the item delegate methods. An exception is thrown if you are using the matrix delegate methods.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setRowHeight:` (page 445)

Declared In
`NSBrowser.h`

scrollColumnsLeftBy:

Scrolls columns left by the specified number of columns.

```
- (void)scrollColumnsLeftBy:(NSInteger)shiftAmount
```

Parameters
shiftAmount
The number of columns by which to scroll the browser.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBrowser.h`

scrollColumnsRightBy:

Scrolls columns right by the specified number of columns.

```
- (void)scrollColumnsRightBy:(NSInteger)shiftAmount
```

Parameters
shiftAmount
The number of columns by which to scroll the browser.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBrowser.h`

scrollColumnToVisible:

Scrolls to make the specified column visible.

```
- (void)scrollColumnToVisible:(NSInteger)column
```

Parameters

column

The index of the column to scroll to.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `scrollRowToVisible:inColumn:` (page 429)

Declared In
`NSBrowser.h`

scrollRowToVisible:inColumn:

Scrolls the specified row to be visible within the specified column.

```
- (void)scrollRowToVisible:(NSInteger)row
    inColumn:(NSInteger)column
```

Parameters

row

The index of the row to scroll.

column

The index of the column containing the row to scroll.

Discussion
The row's column will not be scrolled to visible via this method. To scroll the column to visible, use `scrollColumnToVisible:` (page 428).

Availability
Available in Mac OS X v10.6 and later.

See Also
- `scrollColumnToVisible:` (page 428)

Declared In
`NSBrowser.h`

scrollViaScroller:

Scrolls columns left or right based on an `NSScroller`. (Deprecated in Mac OS X v10.3. There is no replacement.)

```
- (void)scrollViaScroller:(NSScroller *)sender
```

Parameters

sender

The `NSScroller` object that determines the scrolling of the browser columns.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

See Also
- `updateScroller` (page 450)

Declared In
`NSBrowser.h`

selectAll:

Selects all cells in the last column of the browser.

```
- (void)selectAll:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedCell` (page 430)
- `selectedCells` (page 431)
- `selectedColumn` (page 431)

Declared In
`NSBrowser.h`

selectedCell

Returns the last (rightmost and lowest) selected cell.

```
- (id)selectedCell
```

Return Value
The selected cell (`NSCell`).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectAll:` (page 430)
- `selectedCells` (page 431)

Declared In
`NSBrowser.h`

selectedCellInColumn:

Returns the last (lowest) cell selected in the given column.

```
- (id)selectedCellInColumn:(NSInteger)column
```

Parameters
column
The column whose last selected cell is to be returned.

Return Value
The last (or lowest) selected cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- loadedCellAtRow:column:` (page 421)
- `- selectedRowInColumn:` (page 432)

Related Sample Code
NewsReader

Declared In
`NSBrowser.h`

selectedCells

Returns all cells selected in the rightmost column.

```
- (NSArray *)selectedCells
```

Return Value
An array of `NSCell` objects representing the selected cells in the rightmost browser column.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- selectAll:` (page 430)
- `- selectedCell` (page 430)

Declared In
`NSBrowser.h`

selectedColumn

Returns the index of the last column with a selected item.

```
- (NSInteger)selectedColumn
```

Return Value
The index of the last column with a selected item, or `-1` if there is no column selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- columnOfMatrix:` (page 409)
- `- selectAll:` (page 430)

Related Sample Code
ZipBrowser

Declared In
NSBrowser.h

selectedRowInColumn:

Returns the row index of the selected cell in the specified column.

```
- (NSInteger)selectedRowInColumn:(NSInteger)column
```

Parameters

column
The column index specifying the column for which to return the selected row.

Return Value
The row index of the selected cell in the specified column.

Availability
Available in Mac OS X v10.0 and later.

See Also
- loadedCellAtRow:column: (page 421)
- selectedCellInColumn: (page 430)

Related Sample Code
NewsReader

Declared In
NSBrowser.h

selectedRowIndexesInColumn:

Provides the indexes of the selected rows in a given column of the browser.

```
- (NSIndexSet *)selectedRowIndexesInColumn:(NSInteger)columnIndex
```

Parameters

columnIndex
The column whose selected rows are provided.

Return Value
Rows selected in column *columnIndex*.

Availability
Available in Mac OS X v10.5 and later.

See Also
- selectRowIndexes:inColumn: (page 434)

Declared In
NSBrowser.h

selectionIndexPath

Returns the index path of the item selected in the browser.

```
- (NSIndexPath *)selectionIndexPath
```

Return Value
The index path of the selected item, or `nil` if there is no selection.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setSelectionIndexPath:` (page 445)

Declared In
`NSBrowser.h`

selectionIndexPaths

Returns an array containing the index paths of all items selected in the browser.

```
- (NSArray *)selectionIndexPaths
```

Return Value
The array containing the index paths of the selected items.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setSelectionIndexPaths:` (page 446)

Declared In
`NSBrowser.h`

selectRow:inColumn:

Selects the cell at the specified row and column index.

```
- (void)selectRow:(NSInteger)row inColumn:(NSInteger)column
```

Parameters

row
The row index of the cell to select.

column
The column index of the cell to select.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- loadedCellAtRow:column:` (page 421)

Declared In
NSBrowser.h

selectRowIndexes:inColumn:

Specifies the selected rows in a given column of the browser.

```
- (void)selectRowIndexes:(NSIndexSet *)rowIndexes inColumn:(NSInteger)columnIndex
```

Parameters

rowIndexes
Rows to be selected in column *columnIndex*.

columnIndex
Column in which to select rows *rowIndexes*.

Availability
Available in Mac OS X v10.5 and later.

See Also
- selectedRowIndexesInColumn: (page 432)

Declared In
NSBrowser.h

sendAction

Sends the action message to the target.

```
- (BOOL)sendAction
```

Return Value
YES if successful; NO if no target for the message could be found.

Availability
Available in Mac OS X v10.0 and later.

See Also
- doClick: (page 412)
- doDoubleClick: (page 412)

Declared In
NSBrowser.h

sendsActionOnArrowKeys

Returns whether pressing an arrow key causes an action message to be sent.

```
- (BOOL)sendsActionOnArrowKeys
```

Return Value
NO if pressing an arrow key scrolls the browser; YES if it also sends the action message specified by setAction: (page 828).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setSendsActionOnArrowKeys:` (page 446)

Declared In
`NSBrowser.h`

separatesColumns

Indicates whether columns are separated by bezeled borders.

```
- (BOOL)separatesColumns
```

Return Value
`YES` if the browser's columns are separated by bezeled borders; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setSeparatesColumns:` (page 446)

Declared In
`NSBrowser.h`

setAcceptsArrowKeys:

Specifies whether the browser allows navigation using the arrow keys. (Deprecated in Mac OS X v10.6. There is no replacement.)

```
- (void)setAcceptsArrowKeys:(BOOL)flag
```

Parameters
flag
`YES` to enable the use of the arrow keys for navigating within and between browsers; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `acceptsArrowKeys` (page 404)
- `sendsActionOnArrowKeys` (page 434)

Declared In
`NSBrowser.h`

setAllowsBranchSelection:

Allows the user to select branch items.

```
- (void)setAllowsBranchSelection:(BOOL)flag
```

Parameters

flag

YES if the user can select branch items when multiple selection is enabled; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- allowsBranchSelection (page 404)

Declared In
NSBrowser.h

setAllowsEmptySelection:

Allows the user to select nothing.

```
- (void)setAllowsEmptySelection:(BOOL)flag
```

Parameters

flag

YES if the browser allows an empty selection; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- allowsEmptySelection (page 405)

Declared In
NSBrowser.h

setAllowsMultipleSelection:

Allows the user to select multiple items.

```
- (void)setAllowsMultipleSelection:(BOOL)flag
```

Parameters

flag

YES if the user can select multiple items at once; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- allowsMultipleSelection (page 405)

Declared In
NSBrowser.h

setAllowsTypeSelect:

Allows the browser to accept keystroke-based selection.

```
- (void)setAllowsTypeSelect:(BOOL)allowsTypeSelection
```

Parameters

allowsTypeSelection

YES to allow type selection; NO to disallow it.

Availability
Available in Mac OS X v10.5 and later.

See Also
- allowsTypeSelect (page 405)

Declared In
NSBrowser.h

setAutohidesScroller:

Allows the browser to hide its scroller automatically.

```
- (void)setAutohidesScroller:(BOOL)flag
```

Parameters

flag

If YES, the browser hides its scroller automatically; if NO, it does not.

Availability
Available in Mac OS X v10.6 and later.

See Also
- autohidesScroller (page 406)

Declared In
NSBrowser.h

setBackgroundColor:

Specifies the browser's background color.

```
- (void)setBackgroundColor:(NSColor *)backgroundColor
```

Parameters

backgroundColor

[NSColor clearColor] specifies a transparent background.

Availability
Available in Mac OS X v10.5 and later.

See Also
- backgroundColor (page 406)

Declared In
NSBrowser.h

setCellClass:

Sets the class of the cell to be used by the matrices in the columns of the browser.

```
- (void)setCellClass:(Class)factoryId
```

Parameters

factoryId

The class of NSCell used by the matrices in the columns of the browser. This method creates an instance of the class and calls setCellPrototype: (page 438).

Availability
Available in Mac OS X v10.0 and later.

See Also
+ cellClass (page 403)
- cellPrototype (page 407)
- setCellPrototype: (page 438)

Declared In
NSBrowser.h

setCellPrototype:

Sets the prototype cell for displaying items in the matrices in the columns of the browser.

```
- (void)setCellPrototype:(NSCell *)aCell
```

Parameters

aCell

The prototype NSCell instance.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ cellClass (page 403)
- cellPrototype (page 407)
- setCellClass: (page 438)

Declared In
NSBrowser.h

setColumnResizingType:

Sets the browser's column resizing type.

```
- (void)setColumnResizingType:(NSBrowserColumnResizingType)columnResizingType
```

Parameters

columnResizingType

The column resizing type. Possible values are described in "NSBrowserColumnResizingType" (page 451). The default is NSBrowserAutoColumnResizing (page 452). This setting is persistent.

Availability
Available in Mac OS X v10.3 and later.

See Also
- columnResizingType (page 409)

Declared In
NSBrowser.h

setColumnsAutosaveName:

Sets the name used to automatically save the browser's column configuration.

```
- (void)setColumnsAutosaveName:(NSString *)name
```

Parameters

name

The name used to save the column configuration. If *name* is different from the current name, this method also reads in any column configuration data previously saved under *name* and applies the values to the browser.

Discussion
Column configuration is defined as an array of column content widths. One width is saved for each level the user has reached. That is, the browser saves column width based on depth, not on unique paths. To do more complex column persistence, you should register for NSBrowserColumnConfigurationDidChangeNotification (page 453) and handle persistence yourself. This setting is persistent.

Availability
Available in Mac OS X v10.3 and later.

See Also
- columnsAutosaveName (page 409)
+ removeSavedColumnsWithAutosaveName: (page 403)

Declared In
NSBrowser.h

setDefaultColumnWidth:

Sets the default column width for new browser columns that do not otherwise have an initial width from defaults or the browser's delegate.

```
- (void)setDefaultColumnWidth:(CGFloat)columnWidth
```

Parameters

columnWidth

The default column width to set.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `defaultColumnWidth` (page 410)

Declared In
`NSBrowser.h`

setDelegate:

Sets the browser's delegate.

```
- (void)setDelegate:(id <NSBrowserDelegate>)anObject
```

Parameters

anObject
The object to set at the browser's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `delegate` (page 411)

Declared In
`NSBrowser.h`

setDoubleAction:

Sets the browser's double-click action.

```
- (void)setDoubleAction:(SEL)aSelector
```

Parameters

aSelector
The action method to invoke when the browser is double-clicked.

Discussion
For the method to have any effect, the browser's action and target must be set to the class in which the selector is declared. See *Action Messages* for additional information on action messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `doubleAction` (page 412)
- `sendAction` (page 434)
- `doDoubleClick:` (page 412)

Related Sample Code
NewsReader

Declared In
NSBrowser.h

setDraggingSourceOperationMask:forLocal:

Specifies the drag-operation mask for dragging operations with local or external destinations.

```
- (void)setDraggingSourceOperationMask:(NSDragOperation)dragOperationMask
    forLocal:(BOOL)localDestination
```

Parameters

dragOperationMask

Dragging operation mask to use for either local or external drag operations, as specified by localDestination.

localDestination

Indicates the location of the dragging operation's destination object:

YES for this application; NO for another application.

Discussion

Important: Do not override this method.

Availability
Available in Mac OS X v10.5 and later.

See Also
- draggingSourceOperationMaskForLocal: (page 413)

Declared In
NSBrowser.h

setHasHorizontalScroller:

Sets whether the browser has a scroller to scroll horizontally.

```
- (void)setHasHorizontalScroller:(BOOL)flag
```

Parameters

flag

YES if the browser uses an NSScroller object to scroll horizontally; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- hasHorizontalScroller (page 417)

Declared In
NSBrowser.h

setLastColumn:

Sets the last column.

```
- (void)setLastColumn:(NSInteger)column
```

Parameters

column

The index of the last column.

Availability

Available in Mac OS X v10.0 and later.

See Also

- lastColumn (page 420)

Declared In

NSBrowser.h

setMatrixClass:

Sets the matrix class to be used in the browser's columns.

```
- (void)setMatrixClass:(Class)factoryId
```

Parameters

factoryId

The matrix class (NSMatrix or an NSMatrix subclass) used in the browser's columns.

Availability

Available in Mac OS X v10.0 and later.

See Also

- matrixClass (page 421)

Declared In

NSBrowser.h

setMaxVisibleColumns:

Sets the maximum number of columns that can be displayed.

```
- (void)setMaxVisibleColumns:(NSInteger)columnCount
```

Parameters

columnCount

The maximum number of visible columns.

Availability

Available in Mac OS X v10.0 and later.

See Also

- maxVisibleColumns (page 422)

Declared In
NSBrowser.h

setMinColumnWidth:

Sets the minimum column width.

```
- (void)setMinColumnWidth:(CGFloat)columnWidth
```

Parameters

columnWidth

The minimum column width, in pixels.

Availability
Available in Mac OS X v10.0 and later.

See Also
- minColumnWidth (page 423)

Declared In
NSBrowser.h

setPath:

Sets the path to be displayed by the browser.

```
- (BOOL)setPath:(NSString *)path
```

Parameters

path

The path to display. If *path* is prefixed by the path separator, the path is absolute, containing the full path from the browser's first column. Otherwise, the path is relative, extending the browser's current path starting at the last column.

Return Value
YES if the given path is valid; otherwise, NO.

Discussion
While parsing *path*, the browser compares each component with the entries in the current column. If an exact match is found, the matching entry is selected, and the next component is compared to the next column's entries. If no match is found for a component, the method exits and returns NO; the final path is set to the valid portion of *path*. If each component of *path* specifies a valid branch or leaf in the browser's hierarchy, the method returns YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- path (page 425)
- pathToColumn: (page 425)

Declared In
NSBrowser.h

setPathSeparator:

Sets the path separator.

```
- (void)setPathSeparator:(NSString *)newString
```

Parameters

newString

The new path separator.

Availability
Available in Mac OS X v10.0 and later.

See Also
- pathSeparator (page 425)

Declared In
NSBrowser.h

setPrefersAllColumnUserResizing:

Causes the browser to resize all column simultaneously rather than resize a single column at a time.

```
- (void)setPrefersAllColumnUserResizing:(BOOL)prefersAllColumnResizing
```

Parameters

prefersAllColumnResizing

YES to cause the browser to resize all columns simultaneously; the default is single column resizing (NO). This setting applies only to browsers that allow the user to resize columns (see NSBrowserUserColumnResizing (page 452). Holding down the Option key while resizing switches the type of resizing used. This setting is persistent.

Availability
Available in Mac OS X v10.3 and later.

See Also
- prefersAllColumnUserResizing (page 426)

Declared In
NSBrowser.h

setReusesColumns:

Specifies whether matrices can be reused.

```
- (void)setReusesColumns:(BOOL)flag
```

Parameters

flag

YES to prevent NSMatrix objects from being freed when their columns are unloaded, so they can be reused; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `reusesColumns` (page 427)

Declared In
`NSBrowser.h`

setRowHeight:

Sets the height of the browser's rows to the specified value.

```
- (void)setRowHeight:(CGFloat)height
```

Parameters

height
The new height to set.

Discussion
The value must be greater than 0. Any fractional value will be forced to an integral value for drawing. For variable row height browsers (ones whose delegates implement `browser:heightOfRow:inColumn:` (page 3588)), the row height will be used to draw alternating rows past the last row in each browser column. `rowHeight` and `setRowHeight:` (page 445) are only available when using the item delegate methods. An exception is thrown if you are using the matrix delegate methods.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `rowHeight` (page 427)

Declared In
`NSBrowser.h`

setSelectionIndexPath:

Sets the browser's selection to the item with the specified path.

```
- (void)setSelectionIndexPath:(NSIndexPath *)path
```

Parameters

path
The path of the item to select.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `selectionIndexPath` (page 433)

Declared In
`NSBrowser.h`

setSelectionIndexPaths:

Sets the browser's selection to the items whose index paths are in the specified array.

```
- (void)setSelectionIndexPaths:(NSArray *)paths
```

Parameters

paths

The array containing the index paths of the items to select.

Availability
Available in Mac OS X v10.6 and later.

See Also
- selectionIndexPaths (page 433)

Declared In
NSBrowser.h

setSendsActionOnArrowKeys:

Allows the specified action message to be sent when the user presses an arrow key.

```
- (void)setSendsActionOnArrowKeys:(BOOL)flag
```

Parameters

flag

YES if pressing an arrow key should send the action message specified by setAction: (page 828) in addition to scrolling the browser; NO if it should only scroll the browser.

Availability
Available in Mac OS X v10.0 and later.

See Also
- sendsActionOnArrowKeys (page 434)

Declared In
NSBrowser.h

setSeparatesColumns:

Separates columns with bezeled borders.

```
- (void)setSeparatesColumns:(BOOL)flag
```

Parameters

flag

YES if the browser's columns should be separated by bezeled borders; otherwise, NO. This value is ignored if isTitled (page 419) does not return NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- separatesColumns (page 435)

Declared In
NSBrowser.h

setTakesTitleFromPreviousColumn:

Sets whether the title of a column is set to the string value of the selected cell in the previous column.

```
- (void)setTakesTitleFromPreviousColumn:(BOOL)flag
```

Parameters

flag

YES if the title of a column should be set to the string value of the selected NSCell in the previous column; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- takesTitleFromPreviousColumn (page 448)

Declared In
NSBrowser.h

setTitle:ofColumn:

Sets the title of the given column.

```
- (void)setTitle:(NSString *)aString ofColumn:(NSInteger)column
```

Parameters

aString

The title of the column.

column

The index of the column whose title should be set.

Availability
Available in Mac OS X v10.0 and later.

See Also
- drawTitleOfColumn:inRect: (page 414)
- titleOfColumn: (page 450)

Declared In
NSBrowser.h

setTitled:

Sets columns to display titles.

```
- (void)setTitled:(BOOL)flag
```

Parameters

flag

YES if the columns in a browser display titles; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isTitled (page 419)

Declared In
NSBrowser.h

setWidth:ofColumn:

Sets the width of the specified column.

```
- (void)setWidth:(CGFloat)columnWidth ofColumn:(NSInteger)columnIndex
```

Parameters

columnWidth

The new width of the specified column.

columnIndex

The index of the column for which to set the width.

Discussion
This method can be used to set the initial width of browser columns unless the column sizing is automatic; setWidth:ofColumn: does nothing if columnResizingType (page 409) is NSBrowserAutoColumnResizing (page 452). To set the default width for new columns (that don't otherwise have initial widths from defaults or via the delegate), use a *columnIndex* of –1. A value set for *columnIndex* of –1 is persistent. An NSBrowserColumnConfigurationDidChangeNotification (page 453) notification is posted (not immediately), if necessary, so that the browser can autosave the new column configuration.

Availability
Available in Mac OS X v10.3 and later.

See Also
- widthOfColumn: (page 451)
- browser:shouldSizeColumn:forUserResize:toWidth: (page 3596) (NSBrowserDelegate)

Declared In
NSBrowser.h

takesTitleFromPreviousColumn

Indicates whether a column takes its title from the selected cell in the previous column.

```
- (BOOL)takesTitleFromPreviousColumn
```

Return Value
`YES` if the title of a column is set to the string value of the selected `NSCell` in the previous column; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTakesTitleFromPreviousColumn:` (page 447)

Declared In
`NSBrowser.h`

tile

Adjusts the various subviews of the browser—scrollers, columns, titles, and so on—without redrawing.

```
- (void)tile
```

Discussion
Your code shouldn't send this message. It's invoked any time the appearance of the browser changes.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSBrowser.h`

titleFrameOfColumn:

Returns the bounds of the title frame for the specified column.

```
- (NSRect)titleFrameOfColumn:(NSInteger)column
```

Parameters
column
The index of the column for which to return the title frame.

Return Value
The rectangle specifying the bounds of the column's title frame.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- drawTitleOfColumn:inRect:` (page 414)

Declared In
`NSBrowser.h`

titleHeight

Returns the height of the column titles.

```
- (CGFloat)titleHeight
```

Return Value
The height of the column titles for the browser.

Availability
Available in Mac OS X v10.0 and later.

See Also
- drawTitleOfColumn:inRect: (page 414)

Declared In
NSBrowser.h

titleOfColumn:

Returns the title displayed for the given column.

```
- (NSString *)titleOfColumn:(NSInteger)column
```

Parameters
column
The index of the column for which to get the title.

Return Value
The title of the specified column.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTitle:ofColumn: (page 447)

Declared In
NSBrowser.h

updateScroller

Updates the horizontal scroller to reflect column positions. (Deprecated in Mac OS X v10.3. There is no replacement.)

```
- (void)updateScroller
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

See Also
- scrollViaScroller: (page 429)

Declared In
NSBrowser.h

validateVisibleColumns

Validates the browser's visible columns.

```
- (void)validateVisibleColumns
```

Discussion
This method invokes the delegate method `browser:isColumnValid:` (page 3589)

Availability
Available in Mac OS X v10.0 and later.

See Also
`- numberOfVisibleColumns` (page 424)

Declared In
NSBrowser.h

widthOfColumn:

Returns the width of the specified column.

```
- (CGFloat)widthOfColumn:(NSInteger)column
```

Parameters
column
The index of the column for which to retrieve the width.

Return Value
The width of the column.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setWidth:ofColumn:` (page 448)

Declared In
NSBrowser.h

Constants

NSBrowserColumnResizingType

Types of browser column resizing.

```
enum {
    NSBrowserNoColumnResizing   = 0,
    NSBrowserAutoColumnResizing = 1,
    NSBrowserUserColumnResizing = 2
};
typedef NSUInteger NSBrowserColumnResizingType;
```

Constants

`NSBrowserNoColumnResizing`

Neither `NSBrowser` nor the user can change the column width. The developer must explicitly set all column widths.

Available in Mac OS X v10.3 and later.

Declared in `NSBrowser.h`.

`NSBrowserAutoColumnResizing`

All columns have the same width, calculated using a combination of the minimum column width and maximum number of visible columns settings. The column width changes as the window size changes. The user cannot resize columns.

Available in Mac OS X v10.3 and later.

Declared in `NSBrowser.h`.

`NSBrowserUserColumnResizing`

The developer chooses the initial column widths, but users can resize all columns simultaneously or each column individually.

Available in Mac OS X v10.3 and later.

Declared in `NSBrowser.h`.

Discussion

These constants are used by the `setColumnResizingType:` (page 438) and `columnResizingType` (page 409) methods.

NSBrowserDropOperation

The type used to specify the drop type of a drag-and-drop operation. See `browser:validateDrop:proposedRow:column:dropOperation:` (page 3600) for more information.

```
enum {
   NSBrowserDropOn,
   NSBrowserDropAbove
};
typedef NSUInteger NSBrowserDropOperation;
```

Constants

`NSBrowserDropOn`

The drop occurs at the row to which the item was dragged.

Available in Mac OS X v10.5 and later.

Declared in `NSBrowser.h`.

`NSBrowserDropAbove`

The drop occurs above the row to which the item was dragged.

Available in Mac OS X v10.5 and later.

Declared in `NSBrowser.h`.

Declared In
NSBrowser.h

Application Kit Versions for NSBrowser Functionality

The version of the AppKit.framework containing a specific bug fix or capability.

```
#define NSAppKitVersionNumberWithContinuousScrollingBrowser 680.0
#define NSAppKitVersionNumberWithColumnResizingBrowser      685.0
```

Constants

`NSAppKitVersionNumberWithContinuousScrollingBrowser`

The specific version of the AppKit framework that introduced support the continuous scrolling in NSBrowser. Developers should not need to use this constant unless they are writing applications for Mac OS X v10.3 and earlier.

Available in Mac OS X v10.3 and later.

Declared in `NSBrowser.h`.

`NSAppKitVersionNumberWithColumnResizingBrowser`

The specific version of the AppKit framework that introduced support for resizing individual browser columns. Developers should not need to use this constant unless they are writing applications for Mac OS X v10.3 and earlier.

Available in Mac OS X v10.3 and later.

Declared in `NSBrowser.h`.

Notifications

NSBrowserColumnConfigurationDidChangeNotification

Notifies the delegate when the width of a browser column has changed. The notification object is the browser whose column sizes need to be made persistent. This notification does not contain a *`userInfo`* dictionary. If the user resizes more than one column, a single notification is posted when the user is finished resizing.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- browserColumnConfigurationDidChange:` (page 3602)

Declared In
NSBrowser.h

NSBrowserCell Class Reference

Inherits from	NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSBrowserCell.h
Companion guide	Browsers
Related sample code	NewsReader SimpleCocoaBrowser

Overview

The `NSBrowserCell` class is the subclass of `NSCell` used by default to display data in the columns of an `NSBrowser` object. (Each column contains an `NSMatrix` filled with `NSBrowserCell` objects.)

The `NSBrowserCell` class implements the user interface of `NSBrowser`.

Tasks

Getting Browser Cell Information

`+ branchImage` (page 456)
Returns the default image for branch cells in a browser.

`+ highlightedBranchImage` (page 457)
Returns the default image for branch browser cells that are highlighted.

Configuring Browser Cells

`- image` (page 458)
Returns the receiver's image.

- `setImage:` (page 460)
  Sets the receiver's image, retaining the image.
- `alternateImage` (page 457)
  Returns the receiver's image for the highlighted state.
- `setAlternateImage:` (page 460)
  Sets the receiver's image for the highlighted state, retaining the image.

Managing Browser Cell State

- `reset` (page 459)
  Unhighlights the receiver and unsets its state.
- `set` (page 459)
  Highlights the receiver and sets its state.
- `isLeaf` (page 458)
  Returns whether the receiver is a leaf or a branch cell.
- `setLeaf:` (page 461)
  Sets whether the receiver is a leaf or a branch cell.
- `isLoaded` (page 459)
  Returns a Boolean value indicating whether the cell is ready to display.
- `setLoaded:` (page 461)
  Sets whether the receiver's state has been set and the cell is ready to display.
- `highlightColorInView:` (page 458)
  Returns the highlight color that the receiver wants to display.

Class Methods

branchImage

Returns the default image for branch cells in a browser.

```
+ (NSImage *)branchImage
```

Return Value
The default image used for branch `NSBrowserCell` objects. The default image is a right-pointing triangle.

Discussion
Override this method if you want a different image. To have a branch `NSBrowserCell` with no image (and no space reserved for an image), override this method to return `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `highlightedBranchImage` (page 457)
- `alternateImage` (page 457)
- `setAlternateImage:` (page 460)

Declared In
NSBrowserCell.h

highlightedBranchImage

Returns the default image for branch browser cells that are highlighted.

```
+ (NSImage *)highlightedBranchImage
```

Return Value
The default image used for branch NSBrowserCell objects that are highlighted. This is a lighter version of the image returned by branchImage (page 456).

Discussion
Override this method if you want a different image.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ branchImage (page 456)
- alternateImage (page 457)
- setAlternateImage: (page 460)

Declared In
NSBrowserCell.h

Instance Methods

alternateImage

Returns the receiver's image for the highlighted state.

```
- (NSImage *)alternateImage
```

Return Value
The image used for the browser cell in its highlighted state or nil if no image is set.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setAlternateImage: (page 460)

Declared In
NSBrowserCell.h

highlightColorInView:

Returns the highlight color that the receiver wants to display.

```
- (NSColor *)highlightColorInView:(NSView *)controlView
```

Parameters

controlView

The view for which to return the highlight color.

Return Value

The highlight color.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSBrowserCell.h

image

Returns the receiver's image.

```
- (NSImage *)image
```

Return Value

The image of the receiver or `nil` if no image is set.

Availability

Available in Mac OS X v10.2 and later.

See Also

`- setImage:` (page 460)

Declared In

NSBrowserCell.h

isLeaf

Returns whether the receiver is a leaf or a branch cell.

```
- (BOOL)isLeaf
```

Return Value

`YES` if the receiver is a leaf cell; otherwise `NO`.

Discussion

A branch `NSBrowserCell` has an image near its right edge indicating that more, hierarchically related information is available; when the user selects the cell, the `NSBrowser` displays a new column of `NSBrowserCell` objects. A leaf `NSBrowserCell` has no image, indicating that the user has reached a terminal piece of information; it doesn't point to additional information.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `setLeaf:` (page 461)

Declared In
`NSBrowserCell.h`

isLoaded

Returns a Boolean value indicating whether the cell is ready to display.

```
- (BOOL)isLoaded
```

Return Value
`YES` if the receiver's state has been set and the cell is ready to display; otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setLoaded:` (page 461)

Declared In
`NSBrowserCell.h`

reset

Unhighlights the receiver and unsets its state.

```
- (void)reset
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `set` (page 459)

Declared In
`NSBrowserCell.h`

set

Highlights the receiver and sets its state.

```
- (void)set
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `reset` (page 459)

Declared In
NSBrowserCell.h

setAlternateImage:

Sets the receiver's image for the highlighted state, retaining the image.

```
- (void)setAlternateImage:(NSImage *)newAltImage
```

Parameters

newAltImage

The new image for the browser cell in its highlighted state. If *newAltImage* is `nil`, it removes the alternate image for the receiver. *newAltImage* is drawn vertically centered on the left edge of the browser cell.

Note that *newAltImage* is drawn at the given size of the image. `NSBrowserCell` does not set the size of the image, nor does it clip the drawing of the image. Make sure *newAltImage* is the correct size for drawing in the browser cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alternateImage` (page 457)

Declared In
NSBrowserCell.h

setImage:

Sets the receiver's image, retaining the image.

```
- (void)setImage:(NSImage *)newImage
```

Parameters

newImage

The new image. If *newImage* is `nil`, it removes the image for the receiver. *newImage* is drawn vertically centered on the left edge of the browser cell.

Note that *newImage* is drawn at the given size of the image. `NSBrowserCell` does not set the size of the image, nor does it clip the drawing of the image. Make sure *newImage* is the correct size for drawing in the browser cell.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `image` (page 458)

Declared In
NSBrowserCell.h

setLeaf:

Sets whether the receiver is a leaf or a branch cell.

```
- (void)setLeaf:(BOOL)flag
```

Parameters

flag

`YES` if the receiver is a leaf cell; otherwise `NO`.

Discussion

A branch `NSBrowserCell` has an image near its right edge indicating that more, hierarchically related information is available; when the user selects the cell, the `NSBrowser` displays a new column of `NSBrowserCell` objects. A leaf `NSBrowserCell` has no image, indicating that the user has reached a terminal piece of information; it doesn't point to additional information.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- isLeaf` (page 458)

Related Sample Code

NewsReader

SimpleCocoaBrowser

Declared In

`NSBrowserCell.h`

setLoaded:

Sets whether the receiver's state has been set and the cell is ready to display.

```
- (void)setLoaded:(BOOL)flag
```

Parameters

flag

`YES` if the receiver's state has been set and the cell is ready to display; otherwise `NO`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- isLoaded` (page 459)

Declared In

`NSBrowserCell.h`

NSBundle Additions Reference

Inherits from	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSHelpManager.h AppKit/NSImage.h AppKit/NSNibLoading.h AppKit/NSSound.h
Companion guide	Resource Programming Guide

Overview

The Application Kit extends the behavior of the Foundation framework's NSBundle class to support the loading of specific resource types.

The `NSBundle` additions add support for the following tasks:

- Loading nib files
- Locating image and sound resources
- Accessing context help from a `Help.plist` file

These methods become part of the `NSBundle` class only for those applications that use the Application Kit.

Tasks

Loading Nib Files

`+ loadNibFile:externalNameTable:withZone:` (page 464)
Unarchives the contents of the nib file and links them to objects in your program.

`+ loadNibNamed:owner:` (page 465)
Unarchives the contents of the nib file and links them to a specific owner object.

`- loadNibFile:externalNameTable:withZone:` (page 466)
Unarchives the contents of a nib file located in the receiver's bundle.

Locating Image Resources

- `URLForImageResource:` (page 468)
  Returns the location of the specified image resource as an NSURL.
- `pathForImageResource:` (page 467)
  Returns the location of the specified image resource file.

Accessing Context Help

- `contextHelpForKey:` (page 465)
  Returns the context-sensitive help for the specified key from the bundle's help file.

Locating Sound Resources

- `pathForSoundResource:` (page 467)
  Returns the location of the specified sound resource file.

Class Methods

loadNibFile:externalNameTable:withZone:

Unarchives the contents of the nib file and links them to objects in your program.

```
+ (BOOL)loadNibFile:(NSString *)fileName externalNameTable:(NSDictionary *)context
     withZone:(NSZone *)zone
```

Parameters

fileName
The location of the nib file specified as an absolute path in the file system.

context
A name table whose keys identify objects associated with your program or the nib file. The newly unarchived objects from the nib file use this table to connect to objects in your program. For example, the nib file uses the object associated with the `NSNibOwner` constant as the nib file's owning object. If you associate an empty `NSMutableArray` object with the `NSNibTopLevelObjects` constant, on output, the array contains the top level objects from the nib file. For descriptions of these constants, see *NSNib Class Reference*.

zone
The memory zone in which to allocate the nib file objects.

Return Value
`YES` if the nib file was loaded successfully; otherwise, `NO`.

Discussion
This method is declared in `NSNibLoading.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSNibLoading.h

loadNibNamed:owner:

Unarchives the contents of the nib file and links them to a specific owner object.

```
+ (BOOL)loadNibNamed:(NSString *)aNibName owner:(id)owner
```

Parameters

aNibName

The name of the nib file, which need not include the `.nib` extension. The file name should not include path information. The object in the *owner* parameter determines the location in which to look for the nib file.

owner

The object to assign as the nib FIle's Owner. If the class of this object has an associated bundle, that bundle is searched for the specified nib file; otherwise, this method looks in the main bundle.

Return Value
`YES` if the nib file was loaded successfully; otherwise, `NO`.

Discussion
This method is declared in `NSNibLoading.h`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ bundleForClass:` (NSBundle)

Related Sample Code
NumberInput_IMKit_Sample
OutputBins2PDE
QTAudioExtractionPanel
Reducer
WhackedTV

Declared In
NSNibLoading.h

Instance Methods

contextHelpForKey:

Returns the context-sensitive help for the specified key from the bundle's help file.

```
- (NSAttributedString *)contextHelpForKey:(NSString *)key
```

Parameters

key

A key in your application's `Help.plist` file that identifies the context-sensitive help to return.

Return Value

The help string or `nil` if the application does not have a `Help.plist` file or the file does not contain an entry for the specified *key*.

Discussion

When you build your application, you can merge multiple RTF-based help files together using the `/usr/bin/compileHelp` tool, which then packages your help file information into a property list named `Help.plist`. After placing this property-list file in your application bundle, you can use this method to extract context help information from it. To look up a particular entry, you specify the name of the original RTF help file in the *key* parameter of this method. For example, if your application project contains a help file named `Copy.rtf`, you would retrieve the text from this file by passing the value `@"Copy.rtf"` to the *key* parameter.

This method is declared in `NSHelpManager.h`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- contextHelpForObject:` (page 1317) (NSHelpManager)

Declared In

`NSHelpManager.h`

loadNibFile:externalNameTable:withZone:

Unarchives the contents of a nib file located in the receiver's bundle.

```
- (BOOL)loadNibFile:(NSString *)fileName externalNameTable:(NSDictionary *)context
    withZone:(NSZone *)zone
```

Parameters

fileName

The name of the nib file, which need not include the `.nib` extension.

context

A name table whose keys identify objects associated with your program or the nib file. The newly unarchived objects from the nib file use this table to connect to objects in your program. For example, the nib file uses the object associated with the `NSNibOwner` constant as the nib file's owning object. If you associate an empty `NSMutableArray` object with the `NSNibTopLevelObjects` constant, on output, the array contains the top level objects from the nib file. For descriptions of these constants, see *NSNib Class Reference*.

zone

The memory zone in which to allocate the nib file objects.

Return Value

`YES` if the nib file was loaded successfully; otherwise, `NO`.

Discussion

This method searches the language-specific project (`.lproj`) directories for the specified nib file. If the file is not there, it searches the bundle's `Resources` directory for a nonlocalized version of the file.

This method is declared in `NSNibLoading.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSNibLoading.h`

pathForImageResource:

Returns the location of the specified image resource file.

```
- (NSString *)pathForImageResource:(NSString *)name
```

Parameters

name

The name of the image resource file, without any pathname information. Including a filename extension is optional.

Return Value
The absolute pathname of the resource file or `nil` if the file was not found.

Discussion
Image resources are those files in the bundle that are recognized by the `NSImage` class, including those that can be converted using the Image IO framework.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `pathForResource:ofType:` (NSBundle)
- `URLForImageResource:` (page 468)

Related Sample Code
CompositeLab
FilterDemo
LayerBackedOpenGLView

Declared In
`NSImage.h`

pathForSoundResource:

Returns the location of the specified sound resource file.

```
- (NSString *)pathForSoundResource:(NSString *)name
```

Parameters

name

The name of the sound resource file, without any pathname information. Including a filename extension is optional

Return Value
The absolute pathname of the resource file or `nil` if the file was not found.

Discussion
Sound resources are those files in the bundle that are recognized by the `NSSound` class. The types of sound files can be determined by calling the `soundUnfilteredFileTypes` (page 2467) method of `NSSound`.

This method is declared in `NSSound.h`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- pathForResource:ofType:` (NSBundle)

Declared In
`NSSound.h`

URLForImageResource:

Returns the location of the specified image resource as an NSURL.

```
- (NSURL *)URLForImageResource:(NSString *)name
```

Parameters

name

The name of the image resource file. Including a filename extension is optional.

Return Value
An `NSURL` for the resource file or `nil` if the file was not found.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- pathForImageResource:` (page 467)

Declared In
`NSImage.h`

NSButton Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSButton.h
Companion guide	Button Programming Topics
Related sample code	CocoaSpeechSynthesisExample FunHouse MyPhoto PDF Annotation Editor WhackedTV

Overview

The `NSButton` class is a subclass of `NSControl` that intercepts mouse-down events and sends an action message to a target object when it's clicked or pressed.

The `NSButton` class uses NSButtonCell to implement its user interface.

`NSButton` and `NSMatrix` both provide a control view, which is needed to display an `NSButtonCell` object. However, while `NSMatrix` requires you to access the `NSButtonCell` objects directly, most of the `NSButton` class' methods are "covers" for identically declared methods in `NSButtonCell`. (In other words, the implementation of the `NSButton` method invokes the corresponding `NSButtonCell` method for you, allowing you to be unconcerned with the existence of the `NSButtonCell`.) The only `NSButtonCell` methods that don't have covers relate to the font used to display the key equivalent and to specific methods for highlighting or showing the state of the `NSButton` (these last are usually set together with the `NSButton setButtonType:` (page 482) method).

Tasks

Configuring Buttons

- `setButtonType:` (page 482)
  Sets how the receiver button highlights while pressed and how it shows its state.
- `getPeriodicDelay:interval:` (page 474)
  Returns by reference the delay and interval periods for a continuous button.
- `setPeriodicDelay:interval:` (page 485)
  Sets the message delay and interval periods for a continuous button.
- `alternateTitle` (page 473)
  Returns the title that the button displays when it's in its alternate state.
- `setAlternateTitle:` (page 479)
  Sets the title that appears on the button when it's in its alternate state.
- `attributedTitle` (page 474)
  Returns the title that the button displays in its normal state as an attributed string.
- `setAttributedTitle:` (page 480)
  Sets the string that appears on the button when it's in its normal state to the given attributed string and redraws the button.
- `attributedAlternateTitle` (page 473)
  Returns the title that the button displays when it's in its alternate state as an attributed string.
- `setAttributedAlternateTitle:` (page 480)
  Sets the title that appears on the button when it's in its alternate state to the given attributed string.
- `title` (page 490)
  Returns the title displayed on the button when it's in its normal state.
- `setTitle:` (page 487)
  Sets the title displayed by the receiver when in its normal state and, if necessary, redraws the button's contents.
- `setTitleWithMnemonic:` (page 488)
  Sets the title of a button with a character denoting an access key.
- `setSound:` (page 486)
  Sets the sound played when the user presses the button.
- `sound` (page 489)
  Returns the sound that's played when the user presses the button.

Configuring Button Images

- `image` (page 475)
  Returns the image that appears on the receiver when it's in its normal state.
- `setImage:` (page 482)
  Sets the receiver's image and redraws the button.
- `alternateImage` (page 472)
  Returns the image that appears on the button when it's in its alternate state.

- `setAlternateImage:` (page 479)
  Sets the image displayed by the button when it's in its alternate state and, if necessary, redraws the contents of the button.
- `imagePosition` (page 476)
  Returns the position of the receiver's image relative to its title.
- `setImagePosition:` (page 483)
  Sets the position of the button's image relative to its title.
- `isBordered` (page 476)
  Returns a Boolean value indicating whether the button has a border.
- `setBordered:` (page 481)
  Sets whether the receiver has a bezeled border.
- `isTransparent` (page 477)
  Returns a Boolean value indicating whether the button is transparent.
- `setTransparent:` (page 488)
  Sets whether the receiver is transparent and redraws the receiver if necessary.
- `bezelStyle` (page 474)
  Returns the appearance of the receiver's border.
- `setBezelStyle:` (page 481)
  Sets the appearance of the border, if the receiver has one.
- `showsBorderOnlyWhileMouseInside` (page 489)
  Returns a Boolean value indicating whether the button displays its border only when the cursor is over it.
- `setShowsBorderOnlyWhileMouseInside:` (page 485)
  Sets whether the receiver's border is displayed only when the cursor is over the button.

Managing Button State

- `allowsMixedState` (page 472)
  Returns a Boolean value indicating whether the button allows a mixed state.
- `setAllowsMixedState:` (page 478)
  Sets whether the button allows a mixed state.
- `state` (page 489)
  Returns the receiver's state.
- `setState:` (page 486)
  Sets the cell's state to the specified value.
- `setNextState` (page 484)
  Sets the receiver to its next state.
- `highlight:` (page 475)
  Highlights (or unhighlights) the receiver.

Accessing Key Equivalents

- `keyEquivalent` (page 477)
  Returns the key-equivalent character of the receiver.

- `setKeyEquivalent:` (page 483)
  Sets the key equivalent character of the receiver to the given character.
- `keyEquivalentModifierMask` (page 477)
  Returns the mask specifying the modifier keys for the receiver's key equivalent.
- `setKeyEquivalentModifierMask:` (page 484)
  Sets the mask indicating the modifier keys used by the receiver's key equivalent.

Handling Keyboard Events

- `performKeyEquivalent:` (page 478)
  Checks the button's key equivalent against the specified event and, if they match, simulates the button being clicked.

Instance Methods

allowsMixedState

Returns a Boolean value indicating whether the button allows a mixed state.

```
- (BOOL)allowsMixedState
```

Return Value
`YES` if the receiver has three states: on, off, and mixed. `NO` if the receiver has two states: on and off. The default is `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAllowsMixedState:` (page 478)
- `setNextState` (page 484)

Declared In
`NSButton.h`

alternateImage

Returns the image that appears on the button when it's in its alternate state.

```
- (NSImage *)alternateImage
```

Return Value
The image displayed by the button when it's in its alternate state, or `nil` if there is no alternate image. Note that some button types don't display an alternate image. Buttons don't display images by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlternateImage:` (page 479)
- `image` (page 475)
- `imagePosition` (page 476)
- `keyEquivalent` (page 477)
- `setButtonType:` (page 482)

Declared In
`NSButton.h`

alternateTitle

Returns the title that the button displays when it's in its alternate state.

```
- (NSString *)alternateTitle
```

Return Value
The string that appears on the receiver when it's in its alternate state, or the empty string if the receiver doesn't display an alternate title. By default, a button's alternate title is "Button."

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlternateTitle:` (page 479)
- `attributedAlternateTitle` (page 473)
- `setButtonType:` (page 482)
- `title` (page 490)

Declared In
`NSButton.h`

attributedAlternateTitle

Returns the title that the button displays when it's in its alternate state as an attributed string.

```
- (NSAttributedString *)attributedAlternateTitle
```

Return Value
The string that appears on the receiver when it's in its alternate state, as an `NSAttributedString`, or the empty string if the receiver doesn't display an alternate title. By default, a button's alternate title is "Button."

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAttributedAlternateTitle:` (page 480)
- `attributedTitle` (page 474)
- `setButtonType:` (page 482)

Declared In
NSButton.h

attributedTitle

Returns the title that the button displays in its normal state as an attributed string.

```
- (NSAttributedString *)attributedTitle
```

Return Value
The string that appears on the receiver when it's in its normal state as an `NSAttributedString`, or an empty attributed string if the receiver doesn't display a title.

A button's title is always displayed if the button doesn't use its alternate contents for highlighting or displaying the alternate state. By default, a button's title is "Button."

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAttributedTitle:` (page 480)
- `attributedAlternateTitle` (page 473)
- `setButtonType:` (page 482)

Declared In
NSButton.h

bezelStyle

Returns the appearance of the receiver's border.

```
- (NSBezelStyle)bezelStyle
```

Return Value
The bezel style of the button. See the "Constants" (page 521) section of NSButtonCell (page 491) for the list of possible values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBezelStyle:` (page 481)

Declared In
NSButton.h

getPeriodicDelay:interval:

Returns by reference the delay and interval periods for a continuous button.

```
- (void)getPeriodicDelay:(float *)delay interval:(float *)interval
```

Parameters

delay

On return, the amount of time (in seconds) the button will pause before starting to periodically send action messages to the target object. The default delay is taken from a user's default (60 seconds maximum). If the user hasn't specified a default value, *delay* defaults to 0.4 seconds,

interval

On return, the amount of time (in seconds) between each action message that is sent. The default interval is taken from a user's default (60 seconds maximum). If the user hasn't specified a default value, *interval* defaults to 0.075 seconds.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isContinuous` (page 822) (NSControl)

Declared In
`NSButton.h`

highlight:

Highlights (or unhighlights) the receiver.

```
- (void)highlight:(BOOL)flag
```

Parameters

flag

`YES` to highlight the button; `NO` to unhighlight the button. If the current state of the button matches *flag*, no action is taken.

Discussion
Highlighting may involve the button appearing "pushed in" to the screen, displaying its alternate title or image, or causing the button to appear to be "lit."

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setButtonType:` (page 482)

Declared In
`NSButton.h`

image

Returns the image that appears on the receiver when it's in its normal state.

```
- (NSImage *)image
```

Return Value
The image displayed by the button when it's in its normal state, or `nil` if there is no such image. This image is always displayed on a button that doesn't change its contents when highlighting or showing its alternate state. Buttons don't display images by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setImage:` (page 482)
- `alternateImage` (page 472)
- `setButtonType:` (page 482)

Declared In
`NSButton.h`

imagePosition

Returns the position of the receiver's image relative to its title.

```
- (NSCellImagePosition)imagePosition
```

Return Value
The position of the button's image. This is one of the image positions described in the "Constants" (page 613) section of NSCell (page 533).

Discussion
If the title is above, below, or overlapping the image, or if there is no image, the text is horizontally centered within the button.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setImagePosition:` (page 483)
- `setButtonType:` (page 482)
- `setImage:` (page 482)
- `setTitle:` (page 487)

Declared In
`NSButton.h`

isBordered

Returns a Boolean value indicating whether the button has a border.

```
- (BOOL)isBordered
```

Return Value
`YES` if the receiver has a border, `NO` otherwise. A button's border isn't the single line of most other controls' borders—instead, it's a raised bezel. By default, buttons are bordered.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBordered:` (page 481)

Declared In
NSButton.h

isTransparent

Returns a Boolean value indicating whether the button is transparent.

```
- (BOOL)isTransparent
```

Return Value
YES if the receiver is transparent, NO otherwise. A transparent button never draws itself, but it receives mouse-down events and tracks the mouse properly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTransparent: (page 488)

Declared In
NSButton.h

keyEquivalent

Returns the key-equivalent character of the receiver.

```
- (NSString *)keyEquivalent
```

Return Value
The button's key equivalent, or the empty string if one hasn't been defined. Buttons don't have a default key equivalent.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setKeyEquivalent: (page 483)
- performKeyEquivalent: (page 478)
- keyEquivalentFont (page 504) (NSButtonCell)

Declared In
NSButton.h

keyEquivalentModifierMask

Returns the mask specifying the modifier keys for the receiver's key equivalent.

```
- (NSUInteger)keyEquivalentModifierMask
```

Return Value
The mask specifying the modifier keys that are applied to the button's key equivalent. Mask bits are defined in `NSEvent.h`. The only mask bits relevant in button key-equivalent modifier masks are `NSControlKeyMask`, `NSAlternateKeyMask`, and `NSCommandKeyMask`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setKeyEquivalentModifierMask:` (page 484)
- `keyEquivalent` (page 477)

Declared In
`NSButton.h`

performKeyEquivalent:

Checks the button's key equivalent against the specified event and, if they match, simulates the button being clicked.

```
- (BOOL)performKeyEquivalent:(NSEvent *)anEvent
```

Parameters
anEvent
The event containing the key equivalent.

Return Value
`YES` if the key equivalent in anEvent matches the button's key equivalent; `NO` if it does not. This method also returns `NO` if he receiver is blocked by a modal panel or the button is disabled.

Discussion
If the character in *anEvent* matches the receiver's key equivalent, and the modifier flags in *anEvent* match the key-equivalent modifier mask, `performKeyEquivalent:` simulates the user clicking the button and returning `YES`. Otherwise, `performKeyEquivalent:` does nothing and returns `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `keyEquivalent` (page 477)
- `keyEquivalentModifierMask` (page 477)

Declared In
`NSButton.h`

setAllowsMixedState:

Sets whether the button allows a mixed state.

```
- (void)setAllowsMixedState:(BOOL)flag
```

Parameters

flag

YES to indicate that the receiver has three states: on, off, and mixed. If flag is NO, the receiver has two states: on and off.

Availability
Available in Mac OS X v10.0 and later.

See Also
- allowsMixedState (page 472)
- setNextState (page 484)

Related Sample Code
EnhancedAudioBurn
Quartz Composer WWDC 2005 TextEdit

Declared In
NSButton.h

setAlternateImage:

Sets the image displayed by the button when it's in its alternate state and, if necessary, redraws the contents of the button.

```
- (void)setAlternateImage:(NSImage *)image
```

Parameters

image

The image that appears on the receiver when it's in its alternate state. Note that some button types don't display an alternate image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- alternateImage (page 472)
- setButtonType: (page 482)
- setImage: (page 482)

Declared In
NSButton.h

setAlternateTitle:

Sets the title that appears on the button when it's in its alternate state.

```
- (void)setAlternateTitle:(NSString *)aString
```

Parameters

aString

The string to set as the button's alternate title. Note that some button types don't display an alternate title.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alternateTitle` (page 473)
- `setTitle:` (page 487)
- `setTitleWithMnemonic:` (page 488)
- `setButtonType:` (page 482)
- `setFont:` (page 511) (`NSButtonCell`)

Declared In
`NSButton.h`

setAttributedAlternateTitle:

Sets the title that appears on the button when it's in its alternate state to the given attributed string.

```
- (void)setAttributedAlternateTitle:(NSAttributedString *)aString
```

Parameters

aString

The attributed string to set as the button's alternate title. Note that some button types don't display an alternate title.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `attributedAlternateTitle` (page 473)
- `setAttributedTitle:` (page 480)
- `setButtonType:` (page 482)
- `setFont:` (page 511) (`NSButtonCell`)

Declared In
`NSButton.h`

setAttributedTitle:

Sets the string that appears on the button when it's in its normal state to the given attributed string and redraws the button.

```
- (void)setAttributedTitle:(NSAttributedString *)aString
```

Parameters

aString

The attributed string to set as the button's title. The title is always shown on buttons that don't use their alternate contents when highlighting or displaying their alternate state.

Discussion

Availability
Available in Mac OS X v10.0 and later.

See Also
- `attributedTitle` (page 474)
- `setAttributedAlternateTitle:` (page 480)
- `setButtonType:` (page 482)
- `setFont:` (page 511) (`NSButtonCell`)

Declared In
`NSButton.h`

setBezelStyle:

Sets the appearance of the border, if the receiver has one.

```
- (void)setBezelStyle:(NSBezelStyle)bezelStyle
```

Parameters

`bezelStyle`

The bezel style of the button. This must be one of the bezel styles described in the "Constants" (page 521) section of NSButtonCell (page 491).

If the button is not bordered, the bezel style is ignored.

Discussion
The button uses shading to look like it's sticking out or pushed in. You can set the shading with the `NSButtonCell` method `setGradientType:` (page 512).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `bezelStyle` (page 474)

Related Sample Code
FunHouse

Declared In
`NSButton.h`

setBordered:

Sets whether the receiver has a bezeled border.

```
- (void)setBordered:(BOOL)flag
```

Parameters

`flag`

`YES` if the receiver should display a border; `NO` if it should not. A button's border is not the single line of most other controls' borders—instead, it's a raised bezel.

Discussion
This method redraws the button if `setBordered:` causes the bordered state to change.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isBordered` (page 476)

Related Sample Code
FunHouse

Declared In
`NSButton.h`

setButtonType:

Sets how the receiver button highlights while pressed and how it shows its state.

```
- (void)setButtonType:(NSButtonType)aType
```

Parameters

aType

A constant specifying the type of the button—one of the constants described in the Constants section of `NSButtonCell`.

Discussion
`setButtonType:` redisplays the button before returning.

The types available are for the most common button types, which are also accessible in Interface Builder. You can configure different behavior with the `NSButtonCell` methods `setHighlightsBy:` (page 512) and `setShowsStateBy:` (page 517).

Note that there is no `-buttonType` method. The set method sets various button properties that together establish the behavior of the type.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlternateImage:` (page 479)
- `setImage:` (page 482)
- `setButtonType:` (page 511) (`NSButtonCell`)

Related Sample Code
FunHouse

Declared In
`NSButton.h`

setImage:

Sets the receiver's image and redraws the button.

```
- (void)setImage:(NSImage *)anImage
```

Parameters

anImage

The button's image. A button's image is displayed when the button is in its normal state, or all the time for a button that doesn't change its contents when highlighting or displaying its alternate state.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `image` (page 475)
- `setImagePosition:` (page 483)
- `setAlternateImage:` (page 479)
- `setButtonType:` (page 482)

Related Sample Code
FunHouse

Declared In
`NSButton.h`

setImagePosition:

Sets the position of the button's image relative to its title.

```
- (void)setImagePosition:(NSCellImagePosition)aPosition
```

Parameters

aPosition

A constant specifying the position of the button's image. See the "Constants" (page 613) section of NSCell (page 533) for a listing of possible values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `imagePosition` (page 476)

Related Sample Code
ButtonMadness
FunHouse

Declared In
`NSButton.h`

setKeyEquivalent:

Sets the key equivalent character of the receiver to the given character.

```
- (void)setKeyEquivalent:(NSString *)charCode
```

Parameters

charCode

The character to set as the button's key equivalent.

Discussion

This method redraws the button's interior if it displays a key equivalent instead of an image. The key equivalent isn't displayed if the image position is set to `NSNoImage`, `NSImageOnly`, or `NSImageOverlaps`; that is, the button must display both its title and its "image" (the key equivalent in this case), and they must not overlap.

To display a key equivalent on a button, set the image and alternate image to `nil`, then set the key equivalent, then set the image position.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `keyEquivalent` (page 477)
- `performKeyEquivalent:` (page 478)
- `setAlternateImage:` (page 479)
- `setImage:` (page 482)
- `setImagePosition:` (page 483)
- `setKeyEquivalentFont:` (page 515) (NSButtonCell)

Declared In

`NSButton.h`

setKeyEquivalentModifierMask:

Sets the mask indicating the modifier keys used by the receiver's key equivalent.

```
- (void)setKeyEquivalentModifierMask:(NSUInteger)mask
```

Parameters

mask

The mask identifying the modifier keys to be applied to the button's key equivalent.

Mask bits are defined in `NSEvent.h`. The only mask bits relevant in button key-equivalent modifier masks are `NSControlKeyMask`, `NSAlternateKeyMask`, and `NSCommandKeyMask`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `keyEquivalentModifierMask` (page 477)
- `setKeyEquivalent:` (page 483)

Declared In

`NSButton.h`

setNextState

Sets the receiver to its next state.

```
- (void)setNextState
```

Discussion
If the button has three states, it cycles through them in this order: on, off, mixed, on, and so forth. If the button has two states, it toggles between them.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsMixedState` (page 472)
- `setAllowsMixedState:` (page 478)

Declared In
`NSButton.h`

setPeriodicDelay:interval:

Sets the message delay and interval periods for a continuous button.

```
- (void)setPeriodicDelay:(float)delay interval:(float)interval
```

Parameters

delay

The amount of time (in seconds) that a continuous button will pause before starting to periodically send action messages to the target object. The maximum allowed value is 60.0 seconds; if a larger value is supplied, it is ignored, and 60.0 seconds is used.

interval

The amount of time (in seconds) between each action message. The maximum value is 60.0 seconds; if a larger value is supplied, it is ignored, and 60.0 seconds is used.

Discussion
The delay and interval values are used if the button is configured (by a `setContinuous:` (page 830) message) to continuously send the action message to the target object while tracking the mouse.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setContinuous:` (page 830) (`NSControl`)

Declared In
`NSButton.h`

setShowsBorderOnlyWhileMouseInside:

Sets whether the receiver's border is displayed only when the cursor is over the button.

```
- (void)setShowsBorderOnlyWhileMouseInside:(BOOL)show
```

Parameters

show

YES to display the border only when the cursor is within the button's border and the button is active. NO, to continue to display the button's border when the cursor is outside the button's bounds.

Discussion

If isBordered (page 476) returns NO, the border is never displayed, regardless of what this method returns.

Availability

Available in Mac OS X v10.0 and later.

See Also

- showsBorderOnlyWhileMouseInside (page 489)

Declared In

NSButton.h

setSound:

Sets the sound played when the user presses the button.

```
- (void)setSound:(NSSound *)aSound
```

Parameters

aSound

The sound that should be played when the user presses the button. The sound is played during a mouse-down event, such as NSLeftMouseDown.

Availability

Available in Mac OS X v10.0 and later.

See Also

- sound (page 489)

Declared In

NSButton.h

setState:

Sets the cell's state to the specified value.

```
- (void)setState:(NSInteger)value
```

Parameters

value

The state of the button. This can be NSOnState, NSOffState,NSMixedState. See the discussion for a more detailed explanation.

Discussion

If necessary, this method also redraws the receiver.

The cell can have two or three states. If it has two, *value* can be NSOffState (the normal or unpressed state) and NSOnState (the alternate or pressed state). If it has three, *value* can be NSOnState (the feature is in effect everywhere), NSOffState (the feature is in effect nowhere), or NSMixedState (the feature is in effect somewhere). Note that if the cell has only two states and *value* is NSMixedState, this method sets the cell's state to NSOnState.

Although using the enumerated constants is preferred, *value* can also be an integer. If the cell has two states, 0 is treated as NSOffState, and a nonzero value is treated as NSOnState. If the cell has three states, 0 is treated as NSOffState; a negative value, as NSMixedState; and a positive value, as NSOnState.

To check whether the button uses the mixed state, use the method allowsMixedState (page 472).

Availability
Available in Mac OS X v10.0 and later.

See Also
- state (page 489)

Related Sample Code
EnhancedAudioBurn
FunHouse
QTAudioContextInsert
Sketch-112
WhackedTV

Declared In
NSButton.h

setTitle:

Sets the title displayed by the receiver when in its normal state and, if necessary, redraws the button's contents.

```
- (void)setTitle:(NSString *)aString
```

Parameters

aString
The string to set as the button's title. This title is always shown on buttons that don't use their alternate contents when highlighting or displaying their alternate state.

Availability
Available in Mac OS X v10.0 and later.

See Also
- title (page 490)
- setAlternateTitle: (page 479)
- setButtonType: (page 482)
- setTitleWithMnemonic: (page 488)
- setFont: (page 511) (NSButtonCell)

Related Sample Code
ExtractMovieAudioToAIFF
FunHouse

QTAudioContextInsert
SharedMemory
WhackedTV

Declared In
NSButton.h

setTitleWithMnemonic:

Sets the title of a button with a character denoting an access key.

```
- (void)setTitleWithMnemonic:(NSString *)aString
```

Discussion
Mnemonics are not supported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

See Also
- title (page 490)
- setAlternateTitle: (page 479)
- setButtonType: (page 482)
- setTitle: (page 487)
- setFont: (page 511) (NSButtonCell)

Declared In
NSButton.h

setTransparent:

Sets whether the receiver is transparent and redraws the receiver if necessary.

```
- (void)setTransparent:(BOOL)flag
```

Parameters
flag
YES if the button is transparent; otherwise NO.

Discussion
A transparent button tracks the mouse and sends its action, but doesn't draw. A transparent button is useful for sensitizing an area on the screen so that an action gets sent to a target when the area receives a mouse click.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isTransparent (page 477)

Declared In
NSButton.h

showsBorderOnlyWhileMouseInside

Returns a Boolean value indicating whether the button displays its border only when the cursor is over it.

```
- (BOOL)showsBorderOnlyWhileMouseInside
```

Return Value
`YES` if the receiver's border is displayed only when the cursor is over the button and the button is active; `NO` if the border is displayed all the time.

By default, this method returns `NO`.

Discussion
If `isBordered` (page 476) returns `NO`, the border is never displayed, regardless of what this method returns.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setShowsBorderOnlyWhileMouseInside:` (page 485)

Declared In
`NSButton.h`

sound

Returns the sound that's played when the user presses the button.

```
- (NSSound *)sound
```

Return Value
The sound played when the user presses the button.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setSound:` (page 486)

Declared In
`NSButton.h`

state

Returns the receiver's state.

```
- (NSInteger)state
```

Return Value
The button's state. A button can have two or three states. If it has two, this value is either `NSOffState` (the normal or unpressed state) or `NSOnState` (the alternate or pressed state). If it has three, this value can be `NSOnState` (the feature is in effect everywhere), `NSOffState` (the feature is in effect nowhere), or `NSMixedState` (the feature is in effect somewhere).

Discussion
To check whether the button uses the mixed state, use the method `allowsMixedState` (page 472).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setState:` (page 486)

Related Sample Code
DatePicker
DragNDropOutlineView
FinalCutPro_AppleEvents
GLUT
WhackedTV

Declared In
`NSButton.h`

title

Returns the title displayed on the button when it's in its normal state.

```
- (NSString *)title
```

Return Value
The title displayed on the receiver when it's in its normal state or the empty string if the button doesn't display a title. This title is always displayed if the button doesn't use its alternate contents for highlighting or displaying the alternate state. By default, a button's title is "Button."

Availability
Available in Mac OS X v10.0 and later.

See Also
`- alternateTitle` (page 473)
`- setButtonType:` (page 482)
`- setTitle:` (page 487)
`- setTitleWithMnemonic:` (page 488)

Related Sample Code
SpeedometerView

Declared In
`NSButton.h`

NSButtonCell Class Reference

Inherits from	NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSButtonCell.h
Companion guide	Button Programming Topics
Related sample code	ButtonMadness DragNDropOutlineView FunkyOverlayWindow MyMediaPlayer QTAudioContextInsert

Overview

The `NSButtonCell` class is a subclass of `NSActionCell` used to implement the user interfaces of push buttons, checkboxes (switches), and radio buttons. It can also be used for any other region of a view that's designed to send a message to a target when clicked. The `NSButton` subclass of `NSControl` uses a single `NSButtonCell`.

The `NSButtonCell` class implements the user interface of `NSButton`.

Setting the integer, float, double, or object value of an `NSButtonCell` object results in a call to `setState:` (page 596) with the value converted to integer. In the case of `setObjectValue:` (page 592), `nil` is equivalent to 0, and a non-`nil` object that doesn't respond to `intValue` (page 565) sets the state to 1. Otherwise, the state is set to the object's `intValue` (page 565). Similarly, querying the integer, float, double, or object value of an `NSButtonCell` returns the current state in the requested representation. In the case of `objectValue` (page 572), this is an `NSNumber` containing `YES` for on, `NO` for off, and integer value -1 for the mixed state.

For more information on the behavior of `NSButtonCell`, see the `NSButton` and `NSMatrix` class specifications, and *Button Programming Topics*.

Exceptions

In its implementation of the `compare:` (page 550) method (declared in `NSCell`), `NSButtonCell` raises an `NSBadComparisonException` if the *`otherCell`* argument is not of the `NSButtonCell` class.

Tasks

Setting Titles

- `alternateMnemonic` (page 496)
  Returns the character in the alternate title that's marked as the "keyboard mnemonic."
- `alternateMnemonicLocation` (page 496)
  Returns an unsigned integer indicating the character in the alternate title that's marked as the "keyboard mnemonic."
- `alternateTitle` (page 497)
  Returns the string displayed by the button when it's in its alternate state.
- `attributedAlternateTitle` (page 497)
  Returns the title displayed by the button when it's in its alternate state, as an attributed string.
- `attributedTitle` (page 498)
  Returns the title displayed by the button when it's in its normal state as an attributed string.
- `setAlternateMnemonicLocation:` (page 507)
  Sets the character in the alternate title that should be the "keyboard mnemonic."
- `setAlternateTitle:` (page 507)
  Sets the title the button displays when it's in its alternate state.
- `setAlternateTitleWithMnemonic:` (page 508)
  Sets the title the button displays when it's in its alternate state to the given string with an embedded mnemonic.
- `setAttributedAlternateTitle:` (page 508)
  Sets the string the button displays when it's in its alternate state to the given attributed string.
- `setAttributedTitle:` (page 509)
  Sets the string the button displays when it's in its normal state to the given attributed string and redraws the button.
- `setFont:` (page 511)
  Sets the font used to display the button's title and alternate title.
- `setTitle:` (page 518)
  Sets the title the button displays when in its normal state and, if necessary, redraws the receiver's contents.
- `setTitleWithMnemonic:` (page 519)
  Sets the title the button displays when it's in its normal state to the given string with an embedded mnemonic.
- `title` (page 521)
  Returns the title displayed on the receiver when it's in its normal state.

Managing Images

- `alternateImage` (page 495)
  Returns the image the button displays in its alternate state.
- `imagePosition` (page 502)
  Returns the position of the receiver's image relative to its title.
- `setAlternateImage:` (page 506)
  Sets the image the button displays in its alternate state and, if necessary, redraws its contents.
- `setImagePosition:` (page 513)
  Sets the position of the receiver's image relative to its title.
- `imageScaling` (page 503)
  Returns the scale factor for the receiver's image.
- `setImageScaling:` (page 514)
  Sets the scale factor for the receiver's image.

Managing the Repeat Interval

- `getPeriodicDelay:interval:` (page 501)
  Returns by reference the delay and interval periods for a continuous button.
- `setPeriodicDelay:interval:` (page 516)
  Sets the message delay and interval for the receiver.

Managing the Key Equivalent

- `keyEquivalent` (page 504)
  Returns the receiver's key-equivalent character.
- `keyEquivalentFont` (page 504)
  Returns the font used to draw the key equivalent.
- `keyEquivalentModifierMask` (page 505)
  Returns the mask identifying the modifier keys for the button's key equivalent.
- `setKeyEquivalent:` (page 514)
  Sets the key equivalent character of the receiver.
- `setKeyEquivalentModifierMask:` (page 516)
  Sets the mask identifying the modifier keys to use with the button's key equivalent.
- `setKeyEquivalentFont:` (page 515)
  Sets the font used to draw the key equivalent and redisplays the receiver if necessary.
- `setKeyEquivalentFont:size:` (page 515)
  Sets by name and size of the font used to draw the key equivalent.

Managing Graphics Attributes

- `backgroundColor` (page 498)
  Returns the background color of the receiver.

- `setBackgroundColor:` (page 510)
  Sets the background color of the receiver.
- `bezelStyle` (page 499)
  Returns the appearance of the receiver's border.
- `setBezelStyle:` (page 510)
  Sets the appearance of the border, if the receiver has one.
- `gradientType` (page 501)
  Returns the gradient of the receiver's border.
- `setGradientType:` (page 512)
  Sets the type of gradient to use for the receiver.
- `imageDimsWhenDisabled` (page 502)
  Returns a Boolean value that indicates whether the receiver's image and text appear "dim" when the receiver is disabled.
- `setImageDimsWhenDisabled:` (page 513)
  Sets whether the receiver's image appears "dim" when the button cell is disabled.
- `isOpaque` (page 503)
  Returns a Boolean value that indicates whether the receiver is opaque.
- `isTransparent` (page 504)
  Returns a Boolean value that indicates whether the receiver is transparent.
- `setTransparent:` (page 519)
  Sets whether the receiver is transparent.
- `showsBorderOnlyWhileMouseInside` (page 520)
  Returns a Boolean value indicating whether the button displays its border only when the cursor is over it.
- `setShowsBorderOnlyWhileMouseInside:` (page 517)
  Sets whether the receiver's border is displayed only when the cursor is over the button.

Displaying the Cell

- `highlightsBy` (page 501)
  Returns flags indicating how the button highlights when it receives a mouse-down event.
- `setHighlightsBy:` (page 512)
  Sets the way the receiver highlights itself while pressed.
- `setShowsStateBy:` (page 517)
  Sets the way the receiver indicates its alternate state.
- `setButtonType:` (page 511)
  Sets how the receiver highlights while pressed and how it shows its state.
- `showsStateBy` (page 520)
  Returns the flags indicating how the button cell shows its alternate state.

Managing the Sound

- `sound` (page 520)
  Returns the sound that's played when the user presses the receiver.

- `setSound:` (page 518)
  Sets the sound that's played when the user presses the receiver.

Handling Events and Action Messages

- `mouseEntered:` (page 505)
  Draws the receiver's border.
- `mouseExited:` (page 506)
  Erases the receiver's border.
- `performClick:` (page 506)
  Simulates the user clicking the receiver with the cursor.

Drawing the Button Content

- `drawBezelWithFrame:inView:` (page 499)
  Draws the border of the button using the current bezel style.
- `drawImage:withFrame:inView:` (page 499)
  Draws the image associated with the button's current state.
- `drawTitle:withFrame:inView:` (page 500)
  Draws the button's title centered vertically in a specified rectangle.

Instance Methods

alternateImage

Returns the image the button displays in its alternate state.

```
- (NSImage *)alternateImage
```

Return Value
The image displayed by the button when it's in its alternate state, or `nil` if there is no alternate image.

Discussion
Note that some button types don't display an alternate image. Buttons don't display images by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlternateImage:` (page 506)
- `imagePosition` (page 502)
- `keyEquivalent` (page 504)
- `setButtonType:` (page 511)
- `image` (page 562) (`NSCell`)

Declared In
NSButtonCell.h

alternateMnemonic

Returns the character in the alternate title that's marked as the "keyboard mnemonic."

```
- (NSString *)alternateMnemonic
```

Return Value
The character in the alternate title (the title displayed on the receiver when it's in its alternate state) marked as the "keyboard mnemonic."

Discussion
Mnemonics are not supported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alternateMnemonicLocation` (page 496)
- `setAlternateTitleWithMnemonic:` (page 508)
- `mnemonic` (page 571) (`NSCell`)

Declared In
NSButtonCell.h

alternateMnemonicLocation

Returns an unsigned integer indicating the character in the alternate title that's marked as the "keyboard mnemonic."

```
- (NSUInteger)alternateMnemonicLocation
```

Return Value
An unsigned integer indicating the character in the alternate title (the title displayed on the receiver when it's in its alternate state) that's marked as the "keyboard mnemonic." If the alternate title doesn't have a keyboard mnemonic, returns `NSNotFound`.

Discussion
Mnemonics are not supported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlternateMnemonicLocation:` (page 507)
- `alternateMnemonic` (page 496)
- `setAlternateTitleWithMnemonic:` (page 508)
- `mnemonicLocation` (page 571) (`NSCell`)

Declared In
NSButtonCell.h

alternateTitle

Returns the string displayed by the button when it's in its alternate state.

```
- (NSString *)alternateTitle
```

Return Value
The string that appears on the button when it's in its alternate state, or the empty string if the receiver doesn't display an alternate title.

Discussion
Note that some button types don't display an alternate title. By default, a button's alternate title is "Button."

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlternateTitle:` (page 507)
- `alternateMnemonic` (page 496)
- `attributedAlternateTitle` (page 497)
- `setButtonType:` (page 511)
- `title` (page 521)

Declared In
NSButtonCell.h

attributedAlternateTitle

Returns the title displayed by the button when it's in its alternate state, as an attributed string.

```
- (NSAttributedString *)attributedAlternateTitle
```

Return Value
The attributed string that appears on the button when it's in its alternate state, or the empty string if the receiver doesn't display an alternate title.

Discussion
Note that some button types don't display an alternate title. By default, a button's alternate title is "Button."

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAttributedAlternateTitle:` (page 508)
- `alternateMnemonic` (page 496)
- `attributedTitle` (page 498)
- `setButtonType:` (page 511)

Declared In
NSButtonCell.h

attributedTitle

Returns the title displayed by the button when it's in its normal state as an attributed string.

```
- (NSAttributedString *)attributedTitle
```

Return Value
The attributes string that appears on the button when it's in its normal state, or an empty attributed string if the receiver doesn't display a title.

Discussion
A button's title is always displayed if the button doesn't use its alternate contents for highlighting or displaying the alternate state. By default, a button's title is "Button."

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAttributedTitle:` (page 509)
- `attributedAlternateTitle` (page 497)
- `setButtonType:` (page 511)
- `mnemonic` (page 571) (`NSCell`)

Declared In
NSButtonCell.h

backgroundColor

Returns the background color of the receiver.

```
- (NSColor *)backgroundColor
```

Return Value
The receiver's background color.

Discussion
The background color is used only when drawing borderless buttons.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setBackgroundColor:` (page 510)

Declared In
NSButtonCell.h

bezelStyle

Returns the appearance of the receiver's border.

```
- (NSBezelStyle)bezelStyle
```

Return Value
A constant specifying the bezel style used by the button. See "Bezel Styles" (page 522) for a list of possible values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setBezelStyle: (page 510)

Declared In
NSButtonCell.h

drawBezelWithFrame:inView:

Draws the border of the button using the current bezel style.

```
- (void)drawBezelWithFrame:(NSRect)frame inView:(NSView *)controlView
```

Parameters

frame
The bounding rectangle of the button.

controlView
The control being drawn.

Discussion
This method is called automatically when the button is redrawn; you should not call it directly.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setBezelStyle: (page 510)

Declared In
NSButtonCell.h

drawImage:withFrame:inView:

Draws the image associated with the button's current state.

```
- (void)drawImage:(NSImage *)image withFrame:(NSRect)frame inView:(NSView
    *)controlView
```

Parameters

image
The image associated with the button's current state.

frame
The bounding rectangle of the button.

controlView
The control being drawn.

Discussion
This method is called automatically when the button is redrawn; you should not call it directly.

You specify the primary and alternate images for the button using Interface Builder.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setAlternateImage:` (page 506)

Declared In
`NSButtonCell.h`

drawTitle:withFrame:inView:

Draws the button's title centered vertically in a specified rectangle.

```
- (NSRect)drawTitle:(NSAttributedString *)title withFrame:(NSRect)frame
    inView:(NSView *)controlView
```

Parameters
title
The title of the button.

frame
The rectangle in which to draw the title.

controlView
The control being drawn.

Return Value
The bounding rectangle for the text of the title.

Discussion
This method is called automatically when the button is redrawn; you should not call it directly.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setAlternateTitle:` (page 507)
- `setAttributedTitle:` (page 509)

Declared In
`NSButtonCell.h`

getPeriodicDelay:interval:

Returns by reference the delay and interval periods for a continuous button.

```
- (void)getPeriodicDelay:(float *)delay interval:(float *)interval
```

Parameters

delay

On return, the amount of time (in seconds) that the button will pause before starting to periodically send action messages to the target object. Default values are taken from the user's defaults (60 seconds maximum); if the user hasn't specified a default value, this defaults to 0.4 seconds.

interval

On return, the amount of time (in seconds) between each action message. Default values are taken from the user's defaults (60 seconds maximum); if the user hasn't specified a default value, this defaults to 0.075 seconds.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isContinuous` (page 566)
- `isContinuous` (page 566) (`NSCell`)

Declared In
`NSButtonCell.h`

gradientType

Returns the gradient of the receiver's border.

```
- (NSGradientType)gradientType
```

Return Value
A constant specifying the gradient used for the button's border. See "Gradient Types" (page 526) for a list of possible values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setGradientType:` (page 512)

Declared In
`NSButtonCell.h`

highlightsBy

Returns flags indicating how the button highlights when it receives a mouse-down event.

```
- (NSInteger)highlightsBy
```

Return Value
The logical OR of flags that indicate the way the receiver highlights when it receivers a mouse-down event. See the "Constants" (page 613) section of NSCell (page 533) for the list of flags.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHighlightsBy:` (page 512)
- `showsStateBy` (page 520)

Related Sample Code
MyMediaPlayer

Declared In
`NSButtonCell.h`

imageDimsWhenDisabled

Returns a Boolean value that indicates whether the receiver's image and text appear "dim" when the receiver is disabled.

```
- (BOOL)imageDimsWhenDisabled
```

Return Value
`YES` if the button's image and text are dimmed when the button is disabled, otherwise `NO`.

Discussion
By default, all button types except `NSSwitchButton` and `NSRadioButton` do dim when disabled. When buttons of type `NSSwitchButton` and `NSRadioButton` are disabled, only the associated text dims.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setButtonType:` (page 511)
- `setImageDimsWhenDisabled:` (page 513)

Declared In
`NSButtonCell.h`

imagePosition

Returns the position of the receiver's image relative to its title.

```
- (NSCellImagePosition)imagePosition
```

Return Value
The position of the button's image. This is one of the image positions described in the "Constants" (page 613) section of NSCell (page 533).

Discussion
If the title is above, below, or overlapping the image, or if there is no image, the text is horizontally centered within the button.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setImagePosition:` (page 513)
- `setButtonType:` (page 511)
- `setTitle:` (page 518)
- `setImage:` (page 589) (`NSCell`)

Declared In
`NSButtonCell.h`

imageScaling

Returns the scale factor for the receiver's image.

```
- (NSImageScaling)imageScaling
```

Return Value
The scale factor for the receiver's image.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSButtonCell.h`

isOpaque

Returns a Boolean value that indicates whether the receiver is opaque.

```
- (BOOL)isOpaque
```

Return Value
`YES` if the receiver draws over every pixel in its frame, otherwise `NO`.

Discussion
A button cell is opaque only if it isn't transparent and if it has a border.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isTransparent` (page 504)
- `setTransparent:` (page 519)

Declared In
`NSButtonCell.h`

isTransparent

Returns a Boolean value that indicates whether the receiver is transparent.

```
- (BOOL)isTransparent
```

Return Value
`YES` if the receiver is transparent, `NO` otherwise.

Discussion
A transparent button never draws itself, but it receives mouse-down events and tracks the mouse properly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTransparent:` (page 519)
- `isOpaque` (page 503)

Declared In
`NSButtonCell.h`

keyEquivalent

Returns the receiver's key-equivalent character.

```
- (NSString *)keyEquivalent
```

Return Value
The string containing the key equivalent character of the button, or the empty string if one hasn't been defined.

Discussion
Buttons don't have a default key equivalent.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setKeyEquivalent:` (page 514)
- `keyEquivalentFont` (page 504)

Declared In
`NSButtonCell.h`

keyEquivalentFont

Returns the font used to draw the key equivalent.

```
- (NSFont *)keyEquivalentFont
```

Return Value
The font object describing the font used to draw the button's key equivalent, or `nil` if the receiver doesn't have a key equivalent.

Discussion
The default font is the same as that used to draw the title.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setKeyEquivalentFont:` (page 515)
- `setKeyEquivalentFont:size:` (page 515)
- `setFont:` (page 511)

Declared In
`NSButtonCell.h`

keyEquivalentModifierMask

Returns the mask identifying the modifier keys for the button's key equivalent.

```
- (NSUInteger)keyEquivalentModifierMask
```

Return Value
A mask indicating the modifier keys that are applied to the receiver's key equivalent.

Mask bits are defined in `NSEvent.h`. The only mask bits relevant in button key-equivalent modifier masks are `NSControlKeyMask`, `NSAlternateKeyMask`, and `NSCommandKeyMask` bits.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setKeyEquivalentModifierMask:` (page 516)
- `keyEquivalent` (page 504)

Declared In
`NSButtonCell.h`

mouseEntered:

Draws the receiver's border.

```
- (void)mouseEntered:(NSEvent *)event
```

Parameters
event
The event object generated by the mouse movement.

Discussion
This method is called only when the cursor moves onto the receiver and `showsBorderOnlyWhileMouseInside` (page 520) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSButtonCell.h

mouseExited:

Erases the receiver's border.

```
- (void)mouseExited:(NSEvent *)event
```

Parameters

event
The event object generated by the mouse movement.

Discussion
This method is called only when the cursor moves off the receiver and `showsBorderOnlyWhileMouseInside` (page 520) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSButtonCell.h

performClick:

Simulates the user clicking the receiver with the cursor.

```
- (void)performClick:(id)sender
```

Parameters

sender
The sender of the message.

Discussion
This method essentially highlights the button, sends the button's action message to the target object, and then unhighlights the button. If an exception is raised while the target object is processing the action message, the button is unhighlighted before the exception is propagated out of `performClick:`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSButtonCell.h

setAlternateImage:

Sets the image the button displays in its alternate state and, if necessary, redraws its contents.

```
- (void)setAlternateImage:(NSImage *)image
```

Parameters

image

The image displayed by the button when it's in its alternate state.

Discussion

Note that some button types don't display an alternate image.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `alternateImage` (page 495)
- `setButtonType:` (page 511)
- `setImage:` (page 589) (`NSCell`)

Declared In

`NSButtonCell.h`

setAlternateMnemonicLocation:

Sets the character in the alternate title that should be the "keyboard mnemonic."

```
- (void)setAlternateMnemonicLocation:(NSUInteger)location
```

Parameters

location

An unsigned integer indicating the character in the alternate title that should be marked as the "keyboard mnemonic." If you don't want the alternate title to have a keyboard mnemonic, specify a location of `NSNotFound`.

Discussion

Mnemonics are not supported in Mac OS X.

The `setAlternateMnemonicLocation:` method doesn't cause the button cell to be redisplayed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `alternateMnemonicLocation` (page 496)
- `setAlternateTitleWithMnemonic:` (page 508)

Declared In

`NSButtonCell.h`

setAlternateTitle:

Sets the title the button displays when it's in its alternate state.

```
- (void)setAlternateTitle:(NSString *)aString
```

Parameters

aString

The string to set as the button's title when it's in its alternate state.

Discussion

Note that some button types don't display an alternate title.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `alternateTitle` (page 497)
- `setAlternateMnemonicLocation:` (page 507)
- `setAlternateTitleWithMnemonic:` (page 508)
- `setTitle:` (page 518)
- `setButtonType:` (page 511)
- `setFont:` (page 511)

Declared In

`NSButtonCell.h`

setAlternateTitleWithMnemonic:

Sets the title the button displays when it's in its alternate state to the given string with an embedded mnemonic.

```
- (void)setAlternateTitleWithMnemonic:(NSString *)aString
```

Parameters

aString

The string to set as the button's alternate title, taking into account the fact that an embedded "&" character is not a literal but instead marks the alternate state's "keyboard mnemonic."

Discussion

Mnemonics are not supported in Mac OS X.

If necessary, `setAlternateTitleWithMnemonic:` redraws the button cell. Note that some button types don't display an alternate title.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setAlternateMnemonicLocation:` (page 507)
- `setTitleWithMnemonic:` (page 519)

Declared In

`NSButtonCell.h`

setAttributedAlternateTitle:

Sets the string the button displays when it's in its alternate state to the given attributed string.

```
- (void)setAttributedAlternateTitle:(NSAttributedString *)aString
```

Parameters

aString

The attributed string to set as the button's alternate title.

Discussion

Note that some button types don't display an alternate title.

Graphics attributes that are set on the cell (`backgroundColor`, `alignment`, `font`, etc.) are overriden when corresponding properties are set for the attributed string.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `attributedAlternateTitle` (page 497)
- `setAlternateMnemonicLocation:` (page 507)
- `setAlternateTitleWithMnemonic:` (page 508)
- `setAttributedTitle:` (page 509)
- `setButtonType:` (page 511)
- `setFont:` (page 511)

Declared In

`NSButtonCell.h`

setAttributedTitle:

Sets the string the button displays when it's in its normal state to the given attributed string and redraws the button.

```
- (void)setAttributedTitle:(NSAttributedString *)aString
```

Parameters

aString

The attributed string to set as the button's title.

Discussion

The title is always shown on buttons that don't use their alternate contents when highlighting or displaying their alternate state.

Graphics attributes configured for the cell (`backgroundColor`, `alignment`, `font`, etc.) are overriden when corresponding properties are set for the attributed string.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `attributedTitle` (page 498)
- `setAttributedAlternateTitle:` (page 508)
- `setButtonType:` (page 511)
- `setFont:` (page 511)
- `setMnemonicLocation:` (page 592) (`NSCell`)

Declared In
NSButtonCell.h

setBackgroundColor:

Sets the background color of the receiver.

```
- (void)setBackgroundColor:(NSColor *)color
```

Parameters

color

The color to use for the receiver's background.

Discussion
The background color is used only when drawing borderless buttons.

Availability
Available in Mac OS X v10.4 and later.

See Also
- backgroundColor (page 498)

Declared In
NSButtonCell.h

setBezelStyle:

Sets the appearance of the border, if the receiver has one.

```
- (void)setBezelStyle:(NSBezelStyle)bezelStyle
```

Parameters

bezelStyle

A constant specifying the bezel style to use for the button. This must be one of the values specified in "Bezel Styles" (page 522).

If the receiver is not bordered, the bezel style is ignored.

Discussion
A button uses shading to look like it's sticking out or pushed in. You can set the shading with setGradientType: (page 512).

Availability
Available in Mac OS X v10.0 and later.

See Also
- bezelStyle (page 499)

Related Sample Code
Sketch+Accessibility

Declared In
NSButtonCell.h

setButtonType:

Sets how the receiver highlights while pressed and how it shows its state.

```
- (void)setButtonType:(NSButtonType)aType
```

Parameters

aType

A constant specifying the type of button. This can be one of the constants defined in "Button Types" (page 524).

Discussion

`setButtonType:` redisplays the receiver before returning.

The types available are for the most common button types, which are also accessible in Interface Builder; you can configure different behavior with the `setHighlightsBy:` (page 512) and `setShowsStateBy:` (page 517) methods.

Note that there is no `-buttonType` method. The set method sets various button properties that together establish the behavior of the type.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setAlternateImage:` (page 506)
- `setImage:` (page 589) (`NSCell`)

Related Sample Code

ButtonMadness

Declared In

`NSButtonCell.h`

setFont:

Sets the font used to display the button's title and alternate title.

```
- (void)setFont:(NSFont *)fontObj
```

Parameters

fontObj

The font object specifying the font to use.

Discussion

This method does nothing if the receiver has no title or alternate title.

If the button cell has a key equivalent, its font is not changed, but the key equivalent's font size is changed to match the new title font.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setKeyEquivalentFont:` (page 515)

- setKeyEquivalentFont:size: (page 515)
- font (page 558) (NSCell)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
Quartz Composer WWDC 2005 TextEdit

Declared In
NSButtonCell.h

setGradientType:

Sets the type of gradient to use for the receiver.

```
- (void)setGradientType:(NSGradientType)gradientType
```

Parameters

gradientType

A constant specifying the gradient to use for the button's border. This can be one of the constants defined in "Gradient Types" (page 526).

Discussion
If the receiver has no border, this method has no effect on its appearance. A concave gradient is darkest in the top-left corner; a convex gradient is darkest in the bottom-right corner. Weak versus strong is how much contrast exists between the colors used in opposite corners.

Note: This method is currently unused by the Application Kit and has no effect.

Availability
Available in Mac OS X v10.0 and later.

See Also
- gradientType (page 501)

Related Sample Code
FunHouse

Declared In
NSButtonCell.h

setHighlightsBy:

Sets the way the receiver highlights itself while pressed.

```
- (void)setHighlightsBy:(NSInteger)aType
```

Parameters

aType

The logical OR of one or more of the cell masks described in the "Constants" (page 613) section of NSCell (page 533).

Discussion
If both `NSChangeGrayCellMask` and `NSChangeBackgroundCellMask` are specified, both are recorded, but which behavior is used depends on the button cell's image. If the button has no image, or if the image has no alpha (transparency) data, `NSChangeGrayCellMask` is used. If the image does have alpha data, `NSChangeBackgroundCellMask` is used; this arrangement allows the color swap of the background to show through the image's transparent pixels.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `highlightsBy` (page 501)
- `setShowsStateBy:` (page 517)

Related Sample Code
MyMediaPlayer

Declared In
`NSButtonCell.h`

setImageDimsWhenDisabled:

Sets whether the receiver's image appears "dim" when the button cell is disabled.

```
- (void)setImageDimsWhenDisabled:(BOOL)flag
```

Parameters

flag

`YES` to indicate that the button's image should dim when the button is disabled.

Discussion
By default, all button types except `NSSwitchButton` and `NSRadioButton` do dim when disabled. When `NSSwitchButton`s and `NSRadioButton`s are disabled, only the associated text dims. The default setting for this condition is reasserted whenever you invoke `setButtonType:` (page 511), so be sure to specify the button cell's type before you invoke `setImageDimsWhenDisabled:`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `imageDimsWhenDisabled` (page 502)

Declared In
`NSButtonCell.h`

setImagePosition:

Sets the position of the receiver's image relative to its title.

```
- (void)setImagePosition:(NSCellImagePosition)aPosition
```

Parameters

aPosition

A constant specifying the position of the button's image. See the "Constants" (page 613) section of NSCell (page 533) for a listing of possible values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- imagePosition (page 502)

Related Sample Code
FunkyOverlayWindow

Declared In
NSButtonCell.h

setImageScaling:

Sets the scale factor for the receiver's image.

```
- (void)setImageScaling:(NSImageScaling)scaling
```

Parameters

scaling

The scale factor for the receiver's image.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSButtonCell.h

setKeyEquivalent:

Sets the key equivalent character of the receiver.

```
- (void)setKeyEquivalent:(NSString *)aKeyEquivalent
```

Parameters

aKeyEquivalent

The key equivalent character.

Discussion
This method redraws the receiver's inside if it displays a key equivalent instead of an image. The key equivalent isn't displayed if the image position is set to NSNoImage, NSImageOnly, or NSImageOverlaps; that is, the button must display both its title and its "image" (the key equivalent in this case), and they must not overlap.

To display a key equivalent on a button, set the image and alternate image to nil, then set the key equivalent, then set the image position.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `keyEquivalent` (page 504)
- `setAlternateImage:` (page 506)
- `setImagePosition:` (page 513)
- `setKeyEquivalentFont:` (page 515)
- `setImage:` (page 589) (NSCell)

Declared In
`NSButtonCell.h`

setKeyEquivalentFont:

Sets the font used to draw the key equivalent and redisplays the receiver if necessary.

```
- (void)setKeyEquivalentFont:(NSFont *)fontObj
```

Parameters

fontObj
The font object specifying the font to use for the receiver's key equivalent.

Discussion
This method does nothing if the receiver doesn't have a key equivalent associated with it.

The default font is the same as that used to draw the title.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `keyEquivalentFont` (page 504)
- `setFont:` (page 511)

Declared In
`NSButtonCell.h`

setKeyEquivalentFont:size:

Sets by name and size of the font used to draw the key equivalent.

```
- (void)setKeyEquivalentFont:(NSString *)fontName size:(CGFloat)fontSize
```

Parameters

fontName
The name of the font to use to draw the key equivalent.

fontSize
The font size to use to draw the key equivalent.

Discussion
This method redisplays the receiver if necessary. It does nothing if the receiver doesn't have a key equivalent associated with it. The default font is the same as that used to draw the title.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `keyEquivalentFont` (page 504)
- `setFont:` (page 511)

Declared In
`NSButtonCell.h`

setKeyEquivalentModifierMask:

Sets the mask identifying the modifier keys to use with the button's key equivalent.

```
- (void)setKeyEquivalentModifierMask:(NSUInteger)mask
```

Parameters

mask

The mask indicating the modifier keys to be applied to the receiver's key equivalent.

Mask bits are defined in `NSEvent.h`. The only mask bits relevant in button key-equivalent modifier masks are `NSControlKeyMask`, `NSAlternateKeyMask`, and `NSCommandKeyMask`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `keyEquivalentModifierMask` (page 505)
- `setKeyEquivalent:` (page 514)

Declared In
`NSButtonCell.h`

setPeriodicDelay:interval:

Sets the message delay and interval for the receiver.

```
- (void)setPeriodicDelay:(float)delay interval:(float)interval
```

Parameters

delay

The amount of time (in seconds) that a continuous button will pause before starting to periodically send action messages to the target object.

The maximum value is 60.0 seconds; if a larger value is supplied, it's ignored, and 60.0 seconds is used.

interval

The amount of time (in seconds) between each action message.

The maximum value is 60.0 seconds; if a larger value is supplied, it's ignored, and 60.0 seconds is used.

Discussion
These values are used if the receiver is configured (by a `setContinuous:` (page 582) message) to continuously send the action message to the target object while tracking the mouse.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setContinuous:` (page 582) (NSCell)

Declared In
`NSButtonCell.h`

setShowsBorderOnlyWhileMouseInside:

Sets whether the receiver's border is displayed only when the cursor is over the button.

```
- (void)setShowsBorderOnlyWhileMouseInside:(BOOL)show
```

Parameters

show

`YES` to display the button's border only when the cursor is within the receiver's border and the button is active. `NO` to continue to display the border when the cursor is outside button's bounds.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `showsBorderOnlyWhileMouseInside` (page 520)

Declared In
`NSButtonCell.h`

setShowsStateBy:

Sets the way the receiver indicates its alternate state.

```
- (void)setShowsStateBy:(NSInteger)aType
```

Parameters

aType

The logical `OR` of one or more of the cell masks described in the "Constants" (page 613) section of NSCell (page 533).

Discussion
If both `NSChangeGrayCellMask` and `NSChangeBackgroundCellMask` are specified, both are recorded, but the actual behavior depends on the button cell's image. If the button has no image, or if the image has no alpha (transparency) data, `NSChangeGrayCellMask` is used. If the image exists and has alpha data, `NSChangeBackgroundCellMask` is used; this arrangement allows the color swap of the background to show through the image's transparent pixels.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHighlightsBy:` (page 512)
- `showsStateBy` (page 520)

Declared In
NSButtonCell.h

setSound:

Sets the sound that's played when the user presses the receiver.

```
- (void)setSound:(NSSound *)aSound
```

Parameters

aSound

The sound to play when the button is pressed.

Discussion
The sound is played during a mouse-down event, such as `NSLeftMouseDown`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sound` (page 520)

Declared In
NSButtonCell.h

setTitle:

Sets the title the button displays when in its normal state and, if necessary, redraws the receiver's contents.

```
- (void)setTitle:(NSString *)aString
```

Parameters

aString

The string to set as the button's title.

Discussion
The title is always shown on buttons that don't use their alternate contents when highlighting or displaying their alternate state.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `title` (page 521)
- `setAlternateTitle:` (page 507)
- `setButtonType:` (page 511)
- `setFont:` (page 511)
- `setTitleWithMnemonic:` (page 519)

Related Sample Code
ButtonMadness

Declared In
NSButtonCell.h

setTitleWithMnemonic:

Sets the title the button displays when it's in its normal state to the given string with an embedded mnemonic.

```
- (void)setTitleWithMnemonic:(NSString *)aString
```

Parameters

aString

The string to set as the button's title, taking into account the fact that an embedded "&" character is not a literal but instead marks the alternate state's "keyboard mnemonic." This title is always shown on buttons that don't use their alternate contents when highlighting or displaying their alternate state.

Discussion
If necessary, `setTitleWithMnemonic:` redraws the button cell. Mnemonics are not supported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlternateTitleWithMnemonic:` (page 508)
- `setTitleWithMnemonic:` (page 599) (`NSCell`)
- `setMnemonicLocation:` (page 592) (`NSCell`)

Declared In
NSButtonCell.h

setTransparent:

Sets whether the receiver is transparent.

```
- (void)setTransparent:(BOOL)flag
```

Parameters

flag

`YES` to make the button cell transparent.

Discussion
This method redraws the receiver if necessary. A transparent button tracks the mouse and sends its action, but doesn't draw. A transparent button is useful for sensitizing an area on the screen so that an action gets sent to a target when the area receives a mouse click.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isTransparent` (page 504)
- `isOpaque` (page 503)

Declared In
NSButtonCell.h

showsBorderOnlyWhileMouseInside

Returns a Boolean value indicating whether the button displays its border only when the cursor is over it.

```
- (BOOL)showsBorderOnlyWhileMouseInside
```

Return Value
YES if the receiver's border is displayed only when the cursor is over the button and the button is active.

Discussion
By default, this method returns NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setShowsBorderOnlyWhileMouseInside: (page 517)

Declared In
NSButtonCell.h

showsStateBy

Returns the flags indicating how the button cell shows its alternate state.

```
- (NSInteger)showsStateBy
```

Return Value
The logical OR of flags that indicate the way the receiver shows its alternate state. See the "Constants" (page 613) section of NSCell (page 533) for the list of flags.

Availability
Available in Mac OS X v10.0 and later.

See Also
- highlightsBy (page 501)
- setShowsStateBy: (page 517)

Declared In
NSButtonCell.h

sound

Returns the sound that's played when the user presses the receiver.

```
- (NSSound *)sound
```

Return Value
The sound played when the receiver is pressed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setSound:` (page 518)

Declared In
`NSButtonCell.h`

title

Returns the title displayed on the receiver when it's in its normal state.

```
- (NSString *)title
```

Return Value
The title displayed by the button in its normal state, or the empty string if the button doesn't display a title.

Discussion
This title is always displayed if the button doesn't use its alternate contents for highlighting or displaying the alternate state. By default, a button's title is "Button."

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTitle:` (page 518)
- `alternateTitle` (page 497)
- `setButtonType:` (page 511)
- `mnemonic` (page 571) (`NSCell`)
- `mnemonicLocation` (page 571) (`NSCell`)

Declared In
`NSButtonCell.h`

Constants

NSBezelStyle

Type to define bezel styles.

```
typedef NSUInteger NSBezelStyle;
```

Discussion
For possible values, see "Bezel Styles" (page 522).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSButtonCell.h`

Bezel Styles

Define the bezel styles used by `bezelStyle` (page 499) and `setBezelStyle:` (page 510).

```
enum {
    NSRoundedBezelStyle          = 1,
    NSRegularSquareBezelStyle    = 2,
    NSThickSquareBezelStyle      = 3,
    NSThickerSquareBezelStyle    = 4,
    NSDisclosureBezelStyle       = 5,
    NSShadowlessSquareBezelStyle = 6,
    NSCircularBezelStyle         = 7,
    NSTexturedSquareBezelStyle   = 8,
    NSHelpButtonBezelStyle       = 9,
    NSSmallSquareBezelStyle      = 10,
    NSTexturedRoundedBezelStyle  = 11,
    NSRoundRectBezelStyle        = 12,
    NSRecessedBezelStyle         = 13,
    NSRoundedDisclosureBezelStyle = 14,
}
```

Constants

`NSRoundedBezelStyle`

A rounded rectangle button, designed for text.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSRegularSquareBezelStyle`

A rectangular button with a 2 point border, designed for icons.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSThickSquareBezelStyle`

A rectangular button with a 3 point border, designed for icons.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSThickerSquareBezelStyle`

A rectangular button with a 4 point border, designed for icons.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSDisclosureBezelStyle`

A bezel style for use with a disclosure triangle.

To create the disclosure triangle, set the button bezel style to `NSDisclosureBezelStyle` and the button type to `NSOnOffButton`.

Available in Mac OS X v10.3 and later.

Declared in `NSButtonCell.h`.

`NSShadowlessSquareBezelStyle`
Similar to `NSRegularSquareBezelStyle`, but has no shadow so you can abut the cells without overlapping shadows.

This style would be used in a tool palette, for example.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSCircularBezelStyle`
A round button with room for a small icon or a single character.

This style has both regular and small variants, but the large variant is available only in gray at this time.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSTexturedSquareBezelStyle`
A bezel style appropriate for use with textured (metal) windows.

Available in Mac OS X v10.3 and later.

Declared in `NSButtonCell.h`.

`NSHelpButtonBezelStyle`
A round button with a question mark providing the standard help button look.

Available in Mac OS X v10.3 and later.

Declared in `NSButtonCell.h`.

`NSSmallSquareBezelStyle`
A simple square bezel style. Buttons using this style can be scaled to any size.

Available in Mac OS X v10.4 and later.

Declared in `NSButtonCell.h`.

`NSTexturedRoundedBezelStyle`
A textured (metal) bezel style similar in appearance to the Finder's action (gear) button.

The height of this button is fixed.

Available in Mac OS X v10.4 and later.

Declared in `NSButtonCell.h`.

`NSRoundRectBezelStyle`
A bezel style that matches the search buttons in Finder and Mail.

Available in Mac OS X v10.4 and later.

Declared in `NSButtonCell.h`.

`NSRecessedBezelStyle`
A bezel style that matches the recessed buttons in Mail, Finder and Safari.

Available in Mac OS X v10.4 and later.

Declared in `NSButtonCell.h`.

`NSRoundedDisclosureBezelStyle`
A bezel style that matches the disclosure style used in the standard Save panel.

Available in Mac OS X v10.4 and later.

Declared in `NSButtonCell.h`.

Discussion
For examples of how these styles are displayed, see *Button Programming Topics*.

Declared In
NSButtonCell.h

NSButtonType

Type to define button types.

```
typedef NSUInteger NSButtonType;
```

Discussion
For possible values, see "Button Types" (page 524).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSButtonCell.h

Button Types

Represent the button types that can be specified using `setButtonType:` (page 511).

```
enum {
    NSMomentaryLightButton   = 0,
    NSPushOnPushOffButton    = 1,
    NSToggleButton           = 2,
    NSSwitchButton           = 3,
    NSRadioButton            = 4,
    NSMomentaryChangeButton  = 5,
    NSOnOffButton            = 6,
    NSMomentaryPushInButton  = 7,
    NSMomentaryPushButton    = 0,
    NSMomentaryLight         = 7
};
```

Constants

`NSMomentaryLightButton`

While the button is held down it's shown as "lit," and also "pushed in" to the screen if the button is bordered.

This type of button is best for simply triggering actions, as it doesn't show its state; it always displays its normal image or title. This option is called "Momentary Light" in Interface Builder's Button Inspector.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSPushOnPushOffButton`

The first click both highlights and causes the button to be "pushed in" if the button is bordered; a second click returns it to its normal state.

This option is called "Push On Push Off" in Interface Builder's Button Inspector.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSToggleButton`

After the first click, the button displays its alternate image or title; a second click returns the button to its normal state.

This option is called “Toggle” in Interface Builder’s Button Inspector.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSSwitchButton`

This style is a variant of `NSToggleButton` that has no border and is used to represent a checkbox.

This type of button is available as a separate Library item in Interface Builder.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSRadioButton`

This style is similar to `NSSwitchButton`, but it used to constrain a selection to a single element from several.

You typically use this type of button in a group formed by an instance of `NSMatrix`. In Interface Builder, a matrix of this type of button is available as a separate Library item.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSMomentaryChangeButton`

While the button is held down, the alternate image and alternate title are displayed.

The normal image and title are displayed when the button isn’t pressed. This option is called “Momentary Change” in Interface Builder’s Button Inspector.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSOnOffButton`

The first click highlights the button; a second click returns it to the normal (unhighlighted) state.

This option is called “On Off” in Interface Builder’s Button Inspector.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSMomentaryPushInButton`

While the button is held down it’s shown as “lit.”

This type of button is best for simply triggering actions, as it doesn’t show its state; it always displays its normal image or title. This option is called “Momentary Push In” in Interface Builder’s Button Inspector.

This button type is the default.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSMomentaryPushButton`

While the button is held down it’s shown as “lit,” and also “pushed in” to the screen if the button is bordered. (Deprecated. Use `NSMomentaryLightButton` instead.)

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSMomentaryLight`

While the button is held down it's shown as "lit." (Deprecated. Use `NSMomentaryPushInButton` instead.)

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

Discussion
For examples of how these types behave, see *Button Programming Topics*.

Declared In
`NSButtonCell.h`

NSGradientType

Type to define gradient types.

```
typedef NSUInteger NSGradientType;
```

Discussion
For possible values, see "Gradient Types" (page 526).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSButtonCell.h`

Gradient Types

Specify the gradients used by `gradientType` (page 501) and `setGradientType:` (page 512).

```
typedef enum _NSGradientType {
    NSGradientNone          = 0,
    NSGradientConcaveWeak   = 1,
    NSGradientConcaveStrong = 2,
    NSGradientConvexWeak    = 3,
    NSGradientConvexStrong  = 4
} NSGradientType;
```

Constants
`NSGradientNone`

There is no gradient, so the button looks flat.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSGradientConcaveWeak`

The top-left corner is light gray, and the bottom-right corner is dark gray, so the button appears to be pushed in.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSGradientConcaveStrong`
: As with `NSGradientConcaveWeak`, the top-left corner is light gray, and the bottom-right corner is dark gray, but the difference between the grays is greater, so the appearance of being pushed in is stronger.

  Available in Mac OS X v10.0 and later.

  Declared in `NSButtonCell.h`.

`NSGradientConvexWeak`
: The top-left corner is dark gray, and the bottom-right corner is light gray, so the button appears to be sticking out.

  Available in Mac OS X v10.0 and later.

  Declared in `NSButtonCell.h`.

`NSGradientConvexStrong`
: As with `NSGradientConvexWeak`, the top-left corner is dark gray, and the bottom-right corner is light gray, but the difference between the grays is greater, so the appearance of sticking out is stronger.

  Available in Mac OS X v10.0 and later.

  Declared in `NSButtonCell.h`.

Declared In
`NSButtonCell.h`

NSCachedImageRep Class Reference

Inherits from	NSImageRep : NSObject
Conforms to	NSCoding (NSImageRep) NSCopying (NSImageRep) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSCachedImageRep.h
Companion guide	Cocoa Drawing Guide

Overview

An `NSCachedImageRep` object store image data in a form that can be readily transferred to the screen. An `NSCachedImageRep` object differs from other image representation objects in that it simply stores the already rendered image, whereas other image representation objects generally have knowledge about how to render the image from source data.

You typically do not use this class directly. Instead, `NSImage` and its other image representation objects create instances of `NSCachedImageRep` as needed to cache versions of the rendered image. This caching speeds up screen-based drawing for existing images during subsequent rendering operations. Cached image representations are also used to capture drawing commands for images created programmatically by locking focus on an image.

Tasks

Initializing an NSCachedImageRep

`- initWithSize:depth:separate:alpha:` (page 530)
Returns an `NSCachedImageRep` object initialized with the specified image characteristics.

`- initWithWindow:rect:` (page 531)
Returns an `NSCachedImageRep` object initialized for drawing in the specified window.

Getting the Representation

- `rect` (page 531)
  Returns the rectangle where the receiver is cached.
- `window` (page 531)
  Returns the window where the receiver is cached.

Instance Methods

initWithSize:depth:separate:alpha:

Returns an `NSCachedImageRep` object initialized with the specified image characteristics. (Deprecated in Mac OS X v10.6.)

```
- (id)initWithSize:(NSSize)size depth:(NSWindowDepth)depth separate:(BOOL)flag
    alpha:(BOOL)alpha
```

Parameters

size
  The size of the image, measured in points.

depth
  The bit depth of the image. Specify 0 if you want the image to be the same depth as the deepest screen on the current system.

flag
  `YES` if the receiver should use a separate offscreen window to store the image; otherwise, `NO` if the receiver should use a shared window.

alpha
  `YES` if the image includes transparency information; otherwise, `NO`.

Return Value
The initialized `NSCachedImageRep` object or `nil` if the object could not be initialized.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `setAlpha:` (page 1420) (NSImageRep)
- `setBitsPerSample:` (page 1421) (NSImageRep)
- `setCacheDepthMatchesImageDepth:` (page 1364) (NSImage)
- `setCachedSeparately:` (page 1365) (NSImage)

Declared In
`NSCachedImageRep.h`

initWithWindow:rect:

Returns an `NSCachedImageRep` object initialized for drawing in the specified window. (Deprecated in Mac OS X v10.6.)

```
- (id)initWithWindow:(NSWindow *)aWindow rect:(NSRect)aRect
```

Parameters

aWindow

The window (typically offscreen) in which the image is to be rendered. The window is retained by the receiver.

aRect

The position and size of the image in the specified window. This rectangle should be specified in the base coordinate system of the window.

Discussion

You must draw the image yourself in the designated part of the window. There are no `NSCachedImageRep` methods for this purpose.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

`- size` (page 1424) (NSImageRep)

Declared In

`NSCachedImageRep.h`

rect

Returns the rectangle where the receiver is cached. (Deprecated in Mac OS X v10.6.)

```
- (NSRect)rect
```

Return Value

The rectangle in the associated offscreen window where the receiver's image is located.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

`- size` (page 1424) (NSImageRep)

Declared In

`NSCachedImageRep.h`

window

Returns the window where the receiver is cached. (Deprecated in Mac OS X v10.6.)

```
- (NSWindow *)window
```

Return Value
The window (typically offscreen) used to store the image.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

Declared In
`NSCachedImageRep.h`

NSCell Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSCell.h
Companion guide	Control and Cell Programming Topics for Cocoa
Related sample code	AnimatedTableView FunHouse PhotoSearch Quartz Composer WWDC 2005 TextEdit VertexPerformanceTest

Overview

The `NSCell` class provides a mechanism for displaying text or images in an `NSView` object without the overhead of a full `NSView` subclass. It's used heavily by most of the `NSControl` classes to implement their internal workings.

Designated Initializers

When subclassing `NSCell` you must implement all of the designated initializers. Those methods are: init, `initWithCoder:`, `initTextCell:` (page 564), and `initImageCell:` (page 563).

Tasks

Initializing a Cell

`- initImageCell:` (page 563)
Returns an `NSCell` object initialized with the specified image and set to have the cell's default menu.

- `initTextCell:` (page 564)
  Returns an NSCell object initialized with the specified string and set to have the cell's default menu.

Managing Cell Values

- `setObjectValue:` (page 592)
  Sets the receiver's object value.
- `objectValue` (page 572)
  Returns the receiver's value as an Objective-C object
- `hasValidObjectValue` (page 560)
  Returns a Boolean value that indicates whether the receiver has a valid object value.
- `setIntValue:` (page 590)
  Sets the value of the receiver using an integer.
- `intValue` (page 565)
  Returns the receiver's value as an integer.
- `setIntegerValue:` (page 590)
  Sets the value of the receiver using an `NSInteger`.
- `integerValue` (page 564)
  Returns the receiver's value as an `NSInteger`.
- `setStringValue:` (page 596)
  Sets the value of the receiver's cell using an `NSString` object.
- `stringValue` (page 605)
  Returns the value of the receiver's cell as an `NSString` object.
- `setDoubleValue:` (page 584)
  Sets the value of the receiver's cell using a double-precision floating-point number.
- `setFloatValue:` (page 587)
  Sets the value of the receiver's cell using a single-precision floating-point number.
- `floatValue` (page 557)
  Returns the value of the receiver's cell as a single-precision floating-point number.
- `doubleValue` (page 552) Available in Mac OS X v10.0 through Mac OS X v10.5
  Returns the value of the receiver's cell as a double-precision floating-point number.

Managing Cell Attributes

- `setCellAttribute:to:` (page 581)
  Sets the value for the specified cell attribute.
- `cellAttribute:` (page 548)
  Returns the value for the specified cell attribute.
- `setType:` (page 599)
  Sets the type of the cell, changing it to a text cell, image cell, or null cell.
- `type` (page 611)
  Returns the type of the receiver

- `setEnabled:` (page 585)
  Sets whether the receiver is enabled or disabled.
- `isEnabled` (page 567)
  Returns a Boolean value that indicates whether the receiver is enabled or disabled.
- `allowsUndo` (page 546)
  Returns a Boolean value that indicates whether the receiver assumes responsibility for undo operations.
- `setAllowsUndo:` (page 579)
  Sets whether the receiver assumes responsibility for undo operations within the cell.

Managing Display Attributes

- `setBezeled:` (page 581)
  Sets whether the receiver draws itself with a bezeled border.
- `isBezeled` (page 566)
  Returns a Boolean value that indicates whether the receiver has a bezeled border.
- `setBordered:` (page 581)
  Sets whether the receiver draws itself outlined with a plain border.
- `isBordered` (page 566)
  Returns a Boolean value that indicates whether the receiver has a plain border.
- `isOpaque` (page 568)
  Returns a Boolean value that indicates whether the receiver is opaque (nontransparent).
- `setControlTint:` (page 583)
  Sets the receiver's control tint.
- `controlTint` (page 551)
  Returns the receiver's control tint.
- `setBackgroundStyle:` (page 580)
  Sets the background style for the receiver.
- `backgroundStyle` (page 547)
  Returns the background style for the receiver.
- `interiorBackgroundStyle` (page 565)
  Returns the interior background style for the receiver.

Managing Cell State

- `allowsMixedState` (page 545)
  Returns a Boolean value that indicates whether the receiver supports three states.
- `nextState` (page 572)
  Returns the receiver's next state.
- `setAllowsMixedState:` (page 578)
  Sets whether the receiver supports three states or just two.
- `setNextState` (page 592)
  Changes the state of the receiver to its next state.

- `setState:` (page 596)
  Sets the receiver's state to the specified value.
- `state` (page 603)
  Returns the receiver's state.

Modifying Textual Attributes

- `setEditable:` (page 584)
  Sets whether the user can edit the receiver's text.
- `isEditable` (page 567)
  Returns a Boolean value that indicates whether the receiver is editable.
- `setSelectable:` (page 594)
  Sets whether text in the receiver can be selected.
- `isSelectable` (page 569)
  Returns a Boolean value that indicates whether the text of the receiver can be selected.
- `setScrollable:` (page 594)
  Sets whether excess text in the receiver is scrolled past the cell's bounds.
- `isScrollable` (page 569)
  Returns a Boolean value that indicates whether the receiver scrolls excess text past the cell's bounds.
- `setAlignment:` (page 577)
  Sets the alignment of text in the receiver.
- `alignment` (page 545)
  Returns the alignment of text in the receiver.
- `setFont:` (page 587)
  Sets the font to use when the receiver displays text.
- `font` (page 558)
  Returns the font used to display text in the receiver
- `lineBreakMode` (page 570)
  Returns the line break mode currently used when drawing text.
- `setLineBreakMode:` (page 591)
  Sets the line break mode to use when drawing text
- `truncatesLastVisibleLine` (page 610)
  Returns a Boolean value indicating whether the receiver truncates and adds the ellipsis character to the last visible line if the text doesn't fit into the cell bounds.
- `setTruncatesLastVisibleLine:` (page 599)
  Sets whether the receiver truncates and adds the ellipsis character to the last visible line if the text doesn't fit into the cell bounds.
- `setWraps:` (page 601)
  Sets whether text in the receiver wraps when its length exceeds the frame of the cell.
- `wraps` (page 612)
  Returns a Boolean value that indicates whether the receiver wraps its text when the text exceeds the borders of the cell.
- `baseWritingDirection` (page 547)
  Returns the initial writing direction used to determine the actual writing direction for text.

- `setBaseWritingDirection:` (page 580)
  Sets the initial writing direction used to determine the actual writing direction for text .
- `setAttributedStringValue:` (page 579)
  Sets the value of the receiver's cell using an attributed string.
- `attributedStringValue` (page 546)
  Returns the value of the receiver's cell as an attributed string using the receiver's formatter object (if one exists).
- `setAllowsEditingTextAttributes:` (page 578)
  Sets whether the receiver allows the user to edit textual attributes of its contents.
- `allowsEditingTextAttributes` (page 545)
  Returns a Boolean value that indicates whether the receiver allows user editing of textual attributes.
- `setImportsGraphics:` (page 589)
  Sets whether the receiver can import images into its text.
- `importsGraphics` (page 563)
  Returns a Boolean value that indicates whether the text of the receiver can contain imported graphics.
- `setUpFieldEditorAttributes:` (page 600)
  Configures the textual and background attributes of the receiver's field editor.
- `title` (page 609)
  Returns the receiver's title.
- `setTitle:` (page 598)
  Sets the title of the receiver.

Managing the Target and Action

- `setAction:` (page 577)
  Sets the cell's action method to the specified selector.
- `action` (page 544)
  Returns the default action-message selector associated with the cell.
- `setTarget:` (page 598)
  Sets the target object to receive action messages.
- `target` (page 608)
  Returns the target object of the receiver.
- `setContinuous:` (page 582)
  Sets whether the receiver's cell sends its action message continuously to its target during mouse tracking.
- `isContinuous` (page 566)
  Returns a Boolean value that indicates whether the receiver's cell sends its action message continuously to its target during mouse tracking.
- `sendActionOn:` (page 576)
  Sets the conditions on which the receiver sends action messages to its target.

Managing the Image

- `setImage:` (page 589)
  Sets the image to be displayed by the receiver.
- `image` (page 562)
  Returns the image displayed by the receiver (if any).

Managing the Tag

- `setTag:` (page 597)
  Sets the tag of the receiver.
- `tag` (page 605)
  Returns the tag identifying the receiver.

Formatting and Validating Data

- `setFormatter:` (page 588)
  Sets the receiver's formatter object.
- `formatter` (page 559)
  Returns the receiver's formatter object.
- `entryType` (page 556) Deprecated in Mac OS X v10.0 and later
  Returns the type of data the user can type into the receiver. (Deprecated. Use a formatter instead—see `setFormatter:` (page 588).)
- `setEntryType:` (page 585) Deprecated in Mac OS X v10.0 and later
  Sets how numeric data is formatted in the receiver and places restrictions on acceptable input. (Deprecated. Use a formatter instead—see `setFormatter:` (page 588).)
- `isEntryAcceptable:` (page 568) Deprecated in Mac OS X v 10.0 and later
  Returns whether a string representing a numeric or date value is formatted in a suitable way for the cell's entry type. (Deprecated. Use `NSFormatter` instead.)
- `setFloatingPointFormat:left:right:` (page 586) Deprecated in Mac OS X v10.0
  Sets the auto-ranging and floating point number format of the receiver's cell. (Deprecated. Use a formatter instead. See `setFormatter:` (page 588))

Managing Menus

+ `defaultMenu` (page 543)
  Returns the default menu for instances of the receiver.
- `setMenu:` (page 591)
  Sets the contextual menu for the cell.
- `menu` (page 570)
  Returns the receiver's contextual menu.
- `menuForEvent:inRect:ofView:` (page 570)
  Returns the menu associated with the receiver and related to the specified event and frame.

Comparing Cells

- `compare:` (page 550)
  Compares the string values of the receiver another cell, disregarding case.

Respond to Keyboard Events

- `acceptsFirstResponder` (page 544)
  Returns a Boolean value that indicates whether the receiver accepts first responder status.
- `setShowsFirstResponder:` (page 595)
  Sets whether the receiver draws some indication of its first responder status.
- `showsFirstResponder` (page 602)
  Returns a Boolean value that indicates whether the receiver should draw some indication of its first responder status.
- `setTitleWithMnemonic:` (page 599)
  Sets the title of the receiver with one character in the string denoted as an access key.
- `mnemonic` (page 571)
  Returns the character in the receiver's title that appears underlined for use as a mnemonic.
- `refusesFirstResponder` (page 573)
  Returns a Boolean value that indicates whether the receiver should not become the first responder.
- `setMnemonicLocation:` (page 592)
  Sets the character of the receiver's title to be used as a mnemonic character.
- `setRefusesFirstResponder:` (page 593)
  Sets whether the receiver should not become the first responder.
- `mnemonicLocation` (page 571)
  Returns the position of the underlined mnemonic character in the receiver's title.
- `performClick:` (page 573)
  Simulates a single mouse click on the receiver.

Deriving Values

- `takeObjectValueFrom:` (page 607)
  Sets the value of the receiver's cell to the object value obtained from the specified object.
- `takeIntegerValueFrom:` (page 606)
  Sets the value of the receiver's cell to an integer value obtained from the specified object.
- `takeIntValueFrom:` (page 607)
  Sets the value of the receiver's cell to an integer value obtained from the specified object.
- `takeStringValueFrom:` (page 608)
  Sets the value of the receiver's cell to the string value obtained from the specified object.
- `takeDoubleValueFrom:` (page 606)
  Sets the value of the receiver's cell to a double-precision floating-point value obtained from the specified object.

- takeFloatValueFrom: (page 606)
  Sets the value of the receiver's cell to a single-precision floating-point value obtained from the specified object.

Representing an Object

- setRepresentedObject: (page 593)
  Sets the object represented by the receiver.
- representedObject (page 574)
  Returns the object the receiver represents.

Tracking the Mouse

- trackMouse:inRect:ofView:untilMouseUp: (page 610)
  Initiates the mouse tracking behavior in a cell.
- startTrackingAt:inView: (page 603)
  Begins tracking mouse events within the receiver.
- continueTracking:at:inView: (page 550)
  Returns a Boolean value that indicates whether mouse tracking should continue in the receiving cell.
- stopTracking:at:inView:mouseIsUp: (page 604)
  Stops tracking mouse events within the receiver.
- mouseDownFlags (page 572)
  Returns the modifier flags for the last (left) mouse-down event.
+ prefersTrackingUntilMouseUp (page 543)
  Returns a Boolean value that indicates whether tracking stops when the cursor leaves the cell.
- getPeriodicDelay:interval: (page 559)
  Returns the initial delay and repeat values for continuous sending of action messages to target objects.

Hit Testing

- hitTestForEvent:inRect:ofView: (page 561)
  Returns hit testing information for the receiver.

Managing the Cursor

- resetCursorRect:inView: (page 574)
  Sets the receiver to show the I-beam cursor while it tracks the mouse.

Handling Keyboard Alternatives

- keyEquivalent (page 569)
  Returns the key equivalent to clicking the cell.

Managing Focus Rings

+ `defaultFocusRingType` (page 543)
  Returns the default type of focus ring for the receiver.
- `setFocusRingType:` (page 587)
  Sets the type of focus ring to be used.
- `focusRingType` (page 558)
  Returns the type of focus ring currently set for the receiver.

Determining Cell Size

- `calcDrawInfo:` (page 548)
  Recalculates the cell geometry.
- `cellSize` (page 549)
  Returns the minimum size needed to display the receiver.
- `cellSizeForBounds:` (page 549)
  Returns the minimum size needed to display the receiver, constraining it to the specified rectangle.
- `drawingRectForBounds:` (page 552)
  Returns the rectangle within which the receiver draws itself
- `imageRectForBounds:` (page 563)
  Returns the rectangle in which the receiver draws its image.
- `titleRectForBounds:` (page 609)
  Returns the rectangle in which the receiver draws its title text.
- `controlSize` (page 551)
  Returns the size of the receiver.
- `setControlSize:` (page 582)
  Sets the size of the receiver.

Drawing and Highlighting

- `drawWithFrame:inView:` (page 554)
  Draws the receiver's border and then draws the interior of the cell.
- `highlightColorWithFrame:inView:` (page 561)
  Returns the color the receiver uses when drawing the selection highlight.
- `drawInteriorWithFrame:inView:` (page 553)
  Draws the interior portion of the receiver, which includes the image or text portion but does not include the border.
- `controlView` (page 552)
  Returns the receiver's control.
- `setControlView:` (page 583)
  Sets the receiver's control view.
- `highlight:withFrame:inView:` (page 560)
  Redraws the receiver with the specified highlight setting.

- `setHighlighted:` (page 588)
  Sets whether the receiver has a highlighted appearance.
- `isHighlighted` (page 568)
  Returns a Boolean value that indicates whether the receiver is highlighted.

Editing and Selecting Text

- `editWithFrame:inView:editor:delegate:event:` (page 555)
  Begins editing of the receiver's text using the specified field editor.
- `selectWithFrame:inView:editor:delegate:start:length:` (page 575)
  Selects the specified text range in the cell's field editor.
- `sendsActionOnEndEditing` (page 576)
  Returns a Boolean value that indicates whether the receiver's `NSControl` object sends its action message whenever the user finishes editing the cell's text.
- `setSendsActionOnEndEditing:` (page 595)
  Sets whether the receiver's NSControl object sends its action message whenever the user finishes editing the cell's text.
- `endEditing:` (page 555)
  Ends the editing of text in the receiver using the specified field editor.
- `wantsNotificationForMarkedText` (page 612)
  Returns a Boolean value that indicates whether the field editor initiated by the receiver should post text change notifications.
- `fieldEditorForView:` (page 557)
  Returns a custom field editor for editing in the view.
- `usesSingleLineMode` (page 612)
  Returns whether the text cell restricts layout and rendering of its content to a single line.
- `setUsesSingleLineMode:` (page 601)
  Sets whether the text cell restricts layout and rendering of its content to a single line.

Managing Expansion Frames

- `expansionFrameWithFrame:inView:` (page 556)
  Returns the expansion cell frame for the receiver.
- `drawWithExpansionFrame:inView:` (page 554)
  Instructs the receiver to draw in an expansion frame.

User Interface Layout Direction

- `setUserInterfaceLayoutDirection:` (page 600)
  Sets the layout direction of the user interface.
- `userInterfaceLayoutDirection` (page 611)
  Returns the layout direction of the user interface.

Class Methods

defaultFocusRingType

Returns the default type of focus ring for the receiver.

```
+ (NSFocusRingType)defaultFocusRingType
```

Return Value
The default type of focus ring for the receiver (one of the values listed in `NSFocusRingType` (page 4009)).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSCell.h`

defaultMenu

Returns the default menu for instances of the receiver.

```
+ (NSMenu *)defaultMenu
```

Return Value
The default menu. The `NSCell` implementation of this method returns `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- menu` (page 570)
`- setMenu:` (page 591)

Declared In
`NSCell.h`

prefersTrackingUntilMouseUp

Returns a Boolean value that indicates whether tracking stops when the cursor leaves the cell.

```
+ (BOOL)prefersTrackingUntilMouseUp
```

Return Value
`YES` if tracking stops when the cursor leaves the cell, otherwise `NO`.

Discussion
The default implementation returns `NO`. Subclasses may override this method to return a different value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `trackMouse:inRect:ofView:untilMouseUp:` (page 610)

Related Sample Code
AnimatedTableView
Cocoa Tips and Tricks
PhotoSearch

Declared In
`NSCell.h`

Instance Methods

acceptsFirstResponder

Returns a Boolean value that indicates whether the receiver accepts first responder status.

```
- (BOOL)acceptsFirstResponder
```

Return Value
`YES` if the receiver can become the first responder, otherwise `NO`.

Discussion
The default value is `YES` if the receiver is enabled. Subclasses may override this method to return a different value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `performClick:` (page 573)
- `setShowsFirstResponder:` (page 595)
- `setTitleWithMnemonic:` (page 599)

Declared In
`NSCell.h`

action

Returns the default action-message selector associated with the cell.

```
- (SEL)action
```

Return Value
The selector associated with the cell. The `NSCell` implementation of this method returns `NULL` by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAction:` (page 577)
- `setTarget:` (page 598)
- `target` (page 608)

Declared In
`NSCell.h`

alignment

Returns the alignment of text in the receiver.

```
- (NSTextAlignment)alignment
```

Return Value
The alignment of text in the receiver (one of the following constants: `NSLeftTextAlignment`, `NSRightTextAlignment`,`NSCenterTextAlignment`, `NSJustifiedTextAlignment`, `NSNaturalTextAlignment`).

Discussion
The default value is `NSNaturalTextAlignment`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlignment:` (page 577)

Declared In
`NSCell.h`

allowsEditingTextAttributes

Returns a Boolean value that indicates whether the receiver allows user editing of textual attributes.

```
- (BOOL)allowsEditingTextAttributes
```

Return Value
`YES` if the receiver allows the user to edit textual attributes of the cell's text, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAllowsEditingTextAttributes:` (page 578)

Declared In
`NSCell.h`

allowsMixedState

Returns a Boolean value that indicates whether the receiver supports three states.

```
- (BOOL)allowsMixedState
```

Return Value
`YES` if the receiver supports all three states (on, off, and mixed), otherwise `NO` (the receiver supports only the on and off states).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `nextState` (page 572)
- `setAllowsMixedState:` (page 578)
- `setNextState` (page 592)

Declared In
`NSCell.h`

allowsUndo

Returns a Boolean value that indicates whether the receiver assumes responsibility for undo operations.

```
- (BOOL)allowsUndo
```

Return Value
`YES` if the receiver handles undo operations, otherwise `NO`.

Discussion
By default, the `NSTextFieldCell` class uses this feature to handle undo operations for edited text. Other controls set a value that is appropriate for their implementation.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setAllowsUndo:` (page 579)

Declared In
`NSCell.h`

attributedStringValue

Returns the value of the receiver's cell as an attributed string using the receiver's formatter object (if one exists).

```
- (NSAttributedString *)attributedStringValue
```

Return Value
The value of the cell interpreted as an attributed string.

Discussion
The textual attributes are the default paragraph style, the receiver's font and alignment, and whether the receiver is enabled and scrollable.

For Mac OS X v10.3 and later: If you use a class that responds to the selector `attributedStringValue` for the object value of a cell, then the cell will use that method to fetch the string to draw rather than using `stringValue` (page 605).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAttributedStringValue:` (page 579)

Related Sample Code
SourceView

Declared In
`NSCell.h`

backgroundStyle

Returns the background style for the receiver.

```
- (NSBackgroundStyle)backgroundStyle
```

Return Value
The background style for the receiver.

Discussion
The background describes the surface the cell is drawn onto in `drawWithFrame:inView:` (page 554). A control typically sets this before it asks the cell to draw. A cell may draw differently based on background characteristics. For example, a tableview drawing a cell in a selected row might call `[cell setBackgroundStyle:NSBackgroundStyleDark]`. A text cell might decide to render its text white as a result. A rating-style level indicator might draw its stars white instead of gray.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCell.h`

baseWritingDirection

Returns the initial writing direction used to determine the actual writing direction for text.

```
- (NSWritingDirection)baseWritingDirection
```

Return Value
The initial writing direction the receiver uses to determine the actual writing direction for text. See `NSWritingDirection` (page 2748) for possible values.

Discussion
The default value is `NSWritingDirectionNatural` (page 2748).

The Text system uses this value as a hint for calculating the actual direction for displaying Unicode characters. You should not need to call this method directly.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setBaseWritingDirection:` (page 580)

Declared In
`NSCell.h`

calcDrawInfo:

Recalculates the cell geometry.

```
- (void)calcDrawInfo:(NSRect)aRect
```

Parameters

aRect

The reference rectangle to use when calculating the cell information.

Discussion
Objects (such as controls) that manage `NSCell` objects generally maintain a flag that informs them if any of their cells have been modified in such a way that the location or size of the cell should be recomputed. If so, `calcSize` (page 816) method of `NSControl` is automatically invoked prior to the display of the cell, and that method invokes the `calcDrawInfo:` method of the cell.

The default implementation of this method does nothing.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cellSize` (page 549)
- `drawingRectForBounds:` (page 552)

Declared In
`NSCell.h`

cellAttribute:

Returns the value for the specified cell attribute.

```
- (NSInteger)cellAttribute:(NSCellAttribute)aParameter
```

Parameters

aParameter

The cell attribute whose value you want to get. Attributes include the receiver's current state and whether it is disabled, editable, or highlighted.

Return Value
The value for the cell attribute specified by *aParameter*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setCellAttribute:to:` (page 581)

Declared In
`NSCell.h`

cellSize

Returns the minimum size needed to display the receiver.

```
- (NSSize)cellSize
```

Return Value
The size of the cell, or the size (10000, 10000) if the receiver is not a text or image cell. If the cell is an image cell but no image has been set, returns `NSZeroSize`.

Discussion
This method takes into account of the size of the image or text within a certain offset determined by the border type of the cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawingRectForBounds:` (page 552)

Related Sample Code
DragNDropOutlineView
EnhancedDataBurn
QTKitMovieShuffler
SourceView
STUCAuthoringDeviceCocoaSample

Declared In
`NSCell.h`

cellSizeForBounds:

Returns the minimum size needed to display the receiver, constraining it to the specified rectangle.

```
- (NSSize)cellSizeForBounds:(NSRect)aRect
```

Parameters
aRect
The size of the cell, or the size of the `aRect` parameter if the cell is not a text or image cell. If the cell is an image cell but no image has been set, returns `NSZeroSize`.

Discussion
This method takes into account of the size of the image or text within a certain offset determined by the border type of the cell. If the receiver is of text type, the text is resized to fit within *aRect* (as much as *aRect* is within the bounds of the cell).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- drawingRectForBounds:` (page 552)

Related Sample Code
Cocoa Tips and Tricks

Declared In
`NSCell.h`

compare:

Compares the string values of the receiver another cell, disregarding case.

```
- (NSComparisonResult)compare:(id)otherCell
```

Parameters

otherCell

The cell to compare against the receiver. This parameter must be of type `NSCell`; if it is not, this method raises `NSBadComparisonException`.

This value must not be `nil`. If the value is `nil`, the behavior is undefined and may change in future versions of Mac OS X.

Return Value
`NSOrderedAscending` if the string value of the receiver precedes the string value of *otherCell* in lexical ordering, `NSOrderedSame` if the string values are equivalent in lexical value, and `NSOrderedDescending` string value of the receiver follows the string value of *otherCell* in lexical ordering.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSCell.h`

continueTracking:at:inView:

Returns a Boolean value that indicates whether mouse tracking should continue in the receiving cell.

```
- (BOOL)continueTracking:(NSPoint)lastPoint at:(NSPoint)currentPoint inView:(NSView
    *)controlView
```

Parameters

lastPoint

Contains either the initial location of the cursor when tracking began or the previous current point.

currentPoint

The current location of the cursor.

controlView

The `NSControl` object managing the receiver.

Return Value
`YES` if mouse tracking should continue, otherwise `NO`.

Discussion
This method is invoked in `trackMouse:inRect:ofView:untilMouseUp:` (page 610). The default implementation returns `YES` if the cell is set to continuously send action messages to its target when the mouse button is down or the mouse is being dragged. Subclasses can override this method to provide more sophisticated tracking behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `startTrackingAt:inView:` (page 603)
- `stopTracking:at:inView:mouseIsUp:` (page 604)

Declared In
`NSCell.h`

controlSize

Returns the size of the receiver.

```
- (NSControlSize)controlSize
```

Return Value
A value that specifies the size of the receiver (for possible values, see "`NSControlSize`" (page 620)).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setControlSize:` (page 582)

Declared In
`NSCell.h`

controlTint

Returns the receiver's control tint.

```
- (NSControlTint)controlTint
```

Return Value
An `NSControlTint` (page 533) value that specifies the tint of the receiver.).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setControlTint:` (page 583)

Declared In
NSCell.h

controlView

Returns the receiver's control.

```
- (NSView *)controlView
```

Return Value
The view (normally an NSControl object) associated with this cell. The default implementation returns nil.

Availability
Available in Mac OS X v10.0 and later.

See Also
- drawWithFrame:inView: (page 554)
- setControlView: (page 583)

Related Sample Code
AnimatedSlider
ClockControl

Declared In
NSCell.h

doubleValue

Returns the value of the receiver's cell as a double-precision floating-point number.

```
- (double)doubleValue
```

Return Value
The value of the cell interpreted as a double-precision floating-point number. If the receiver is not a text-type cell or the cell value is not scannable, returns 0.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setDoubleValue: (page 584)

Declared In
NSCell.h

drawingRectForBounds:

Returns the rectangle within which the receiver draws itself

```
- (NSRect)drawingRectForBounds:(NSRect)theRect
```

Parameters

theRect

The bounding rectangle of the receiver.

Return Value

The rectangle in which the receiver draws itself. This rectangle is slightly inset from the one in `theRect`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- calcSize` (page 816) (`NSControl`)

Related Sample Code

FunHouse

Declared In

`NSCell.h`

drawInteriorWithFrame:inView:

Draws the interior portion of the receiver, which includes the image or text portion but does not include the border.

```
- (void)drawInteriorWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
```

Parameters

cellFrame

The bounding rectangle of the receiver, or a portion of the bounding rectangle.

controlView

The control that manages the cell.

Discussion

Text-type `NSCell` objects display their contents in a rectangle slightly inset from *cellFrame* using a global `NSText` object. Image-type `NSCell` objects display their contents centered within *cellFrame*. If the proper attributes are set, this method also displays the dotted-line rectangle to indicate if the control is the first responder and highlights the cell. This method is invoked from the `drawCellInside:` (page 818) method of `NSControl` to visually update what the cell displays when its contents change. The drawing done by the `NSCell` implementation is minimal and becomes more complex in objects such as `NSButtonCell` and `NSSliderCell`.

This method draws the cell in the currently focused view, which can be different from the *controlView* passed in. Taking advantage of this is not recommended.

Subclasses often override this method to provide more sophisticated drawing of cell contents. Because `drawWithFrame:inView:` (page 554) invokes `drawInteriorWithFrame:inView:` after it draws the cell's border, do not invoke `drawWithFrame:inView:` (page 554) in your override implementation.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- isHighlighted` (page 568)

`- setShowsFirstResponder:` (page 595)

Declared In
NSCell.h

drawWithExpansionFrame:inView:

Instructs the receiver to draw in an expansion frame.

```
- (void)drawWithExpansionFrame:(NSRect)cellFrame inView:(NSView *)view
```

Parameters

cellFrame
The frame in which to draw.

view
The view in which to draw. This view may be different from the original view that the cell appeared in.

Discussion
This method allows the cell to perform custom expansion tool tip drawing. By default, NSCell simply calls drawWithFrame:inView: (page 554).

Availability
Available in Mac OS X v10.5 and later.

See Also
- expansionFrameWithFrame:inView: (page 556)

Declared In
NSCell.h

drawWithFrame:inView:

Draws the receiver's border and then draws the interior of the cell.

```
- (void)drawWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
```

Parameters

cellFrame
The bounding rectangle of the receiver.

controlView
The control that manages the cell.

Discussion
This method draws the cell in the currently focused view, which can be different from the *controlView* passed in. Taking advantage of this behavior is not recommended, however.

Availability
Available in Mac OS X v10.0 and later.

See Also
- drawInteriorWithFrame:inView: (page 553)

Related Sample Code
DragNDropOutlineView

EnhancedDataBurn
QTKitMovieShuffler
SourceView
STUCAuthoringDeviceCocoaSample

Declared In
NSCell.h

editWithFrame:inView:editor:delegate:event:

Begins editing of the receiver's text using the specified field editor.

```
- (void)editWithFrame:(NSRect)aRect inView:(NSView *)controlView editor:(NSText
    *)textObj delegate:(id)anObject event:(NSEvent *)theEvent
```

Parameters

aRect
The bounding rectangle of the cell.

controlView
The control that manages the cell.

textObj
The field editor to use for editing the cell.

anObject
The object to use as a delegate for the field editor (`textObj` parameter). This delegate object receives various `NSText` delegation and notification methods during the course of editing the cell's contents.

theEvent
The `NSLeftMouseDown` event that initiated the editing behavior.

Discussion
If the receiver isn't a text-type `NSCell` object, no editing is performed. Otherwise, the field editor (*textObj*) is sized to *aRect* and its superview is set to *controlView*, so it exactly covers the receiver. The field editor is then activated and editing begins. It's the responsibility of the delegate to end editing when responding to `textShouldEndEditing:`. Upon ending the editing session, the delegate should remove any data from the field editor.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `endEditing:` (page 555)
- `selectWithFrame:inView:editor:delegate:start:length:` (page 575)

Declared In
NSCell.h

endEditing:

Ends the editing of text in the receiver using the specified field editor.

```
- (void)endEditing:(NSText *)textObj
```

Parameters

textObj

The field editor currently handling the editing of the cell's content.

Discussion

Ends any editing of text that began with a call to `editWithFrame:inView:editor:delegate:event:` (page 555) or `selectWithFrame:inView:editor:delegate:start:length:` (page 575).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSCell.h`

entryType

Returns the type of data the user can type into the receiver. (Deprecated in Mac OS X v10.0 and later. Use a formatter instead—see `setFormatter:` (page 588).)

```
- (NSInteger)entryType
```

Return Value

One of the types listed for this method in "`Data Entry Types`" (page 622). If the receiver is not a text-type cell, or if no type has been set, `NSAnyType` is returned.

Availability

Deprecated in Mac OS X v10.0 and later.

See Also

`- setFormatter:` (page 588)

Declared In

`NSCell.h`

expansionFrameWithFrame:inView:

Returns the expansion cell frame for the receiver.

```
- (NSRect)expansionFrameWithFrame:(NSRect)cellFrame inView:(NSView *)view
```

Parameters

cellFrame

The frame for the receiver.

view

The view in which the receiver will be drawn.

Return Value

The expansion cell frame for the receiver. If the frame is not too small, return an empty rect (`NSZeroRect`), and no expansion tool tip view will be shown.

Discussion
This method allows the cell to return an expansion cell frame if *cellFrame* is too small for the entire contents in the view. When the mouse is hovered over the cell in certain controls, the full cell contents are shown in a special floating tool tip view. By default, `NSCell` returns `NSZeroRect`, while some subclasses (such as `NSTextFieldCell`) will return the proper frame when required.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- drawWithExpansionFrame:inView:` (page 554)

Declared In
`NSCell.h`

fieldEditorForView:

Returns a custom field editor for editing in the view.

```
- (NSTextView *)fieldEditorForView:(NSView *)aControlView
```

Parameters
aControlView
The view containing cells that require a custom field editor.

Return Value
A custom field editor. The field editor must have `isFieldEditor` (page 2902) set to `YES`.

Discussion
This is an override point for `NSCell` subclasses designed to use their own custom field editors. This message is sent to the selected cell of *aControlView* using the `NSWindow` method in `fieldEditor:forObject:` (page 3325).

Returning non-`nil` from this method indicates skipping the standard field editor querying processes including `windowWillReturnFieldEditor:toObject:` (page 3940) delegation.

The default implementation returns `nil`.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSCell.h`

floatValue

Returns the value of the receiver's cell as a single-precision floating-point number.

```
- (float)floatValue
```

Return Value
The value of the cell interpreted as a single-precision floating-point number. If the receiver is not a text-type cell or the cell value is not scannable, returns `0`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSCell.h

focusRingType

Returns the type of focus ring currently set for the receiver.

```
- (NSFocusRingType)focusRingType
```

Return Value
The type of focus ring currently set for the receiver (one of the values listed in `NSFocusRingType` (page 4009)).

Discussion
You can disable a view's focus ring drawing by overriding this method so it always returns `NSFocusRingTypeNone`, or by calling `setFocusRingType:` (page 587) with `NSFocusRingTypeNone`. You should only disable a view from drawing its focus ring if you want to draw your own focus ring, or if there isn't sufficient space to display a focus ring in the default location.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setFocusRingType:` (page 587)
+ `defaultFocusRingType` (page 543)

Declared In
NSCell.h

font

Returns the font used to display text in the receiver

```
- (NSFont *)font
```

Return Value
The receiver's current font, or `nil` if the receiver is not a text-type cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFont:` (page 587)

Related Sample Code
FunHouse
PhotoSearch
QTKitMovieShuffler
STUCAuthoringDeviceCocoaSample

Declared In
NSCell.h

formatter

Returns the receiver's formatter object.

```
- (id)formatter
```

Return Value
An object of type NSFormatter used to format the receiver's content.

Discussion
The returned object handles translation of the receiver's contents between its onscreen representation and its object value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setFormatter: (page 588)

Related Sample Code
Sketch+Accessibility

Declared In
NSCell.h

getPeriodicDelay:interval:

Returns the initial delay and repeat values for continuous sending of action messages to target objects.

```
- (void)getPeriodicDelay:(float *)delay interval:(float *)interval
```

Parameters

delay

On input, a pointer to a floating-point variable. On output, the variable contains the current delay (measured in seconds) before messages are sent. This parameter must not be NULL.

interval

On input, a pointer to a floating point variable. On output, the variable contains the interval (measured in seconds) at which messages are sent. This parameter must not be NULL.

Discussion
The default implementation returns a delay of 0.2 and an interval of 0.025 seconds. Subclasses can override this method to supply their own delay and interval values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isContinuous (page 566)
- setContinuous: (page 582)

Declared In
NSCell.h

hasValidObjectValue

Returns a Boolean value that indicates whether the receiver has a valid object value.

```
- (BOOL)hasValidObjectValue
```

Return Value
YES if the cell has a valid object value, otherwise NO.

Discussion
A valid object value is one that the receiver's formatter can "understand." Objects are always assumed to be valid unless they are rejected by the formatter. Invalid objects can still be accepted by the delegate of the receiver's NSControl object (using the control:didFailToFormatString:errorDescription: delegate method).

Availability
Available in Mac OS X v10.0 and later.

See Also
- objectValue (page 572)
- setObjectValue: (page 592)

Declared In
NSCell.h

highlight:withFrame:inView:

Redraws the receiver with the specified highlight setting.

```
- (void)highlight:(BOOL)flag withFrame:(NSRect)cellFrame inView:(NSView *)controlView
```

Parameters

flag
If YES, the cell is redrawn with a highlight; otherwise, if NO, the highlight is removed.

cellFrame
The bounding rectangle of the receiver.

controlView
The control that manages the cell.

Discussion
Note that the NSCell highlighting does not appear when highlighted cells are printed (although instances of NSTextFieldCell, NSButtonCell, and others can print themselves highlighted). Generally, you cannot depend on highlighting being printed because implementations of this method may choose (or not choose) to use transparency.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawWithFrame:inView:` (page 554)
- `isHighlighted` (page 568)

Declared In
`NSCell.h`

highlightColorWithFrame:inView:

Returns the color the receiver uses when drawing the selection highlight.

```
- (NSColor *)highlightColorWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
```

Parameters

cellFrame
The bounding rectangle of the receiver.

controlView
The control that manages the cell.

Return Value
The color the receiver uses when drawing the selection highlight.

Discussion
You should not assume that a cell would necessarily want to draw itself with the value returned from `selectedControlColor` (page 688). A cell may wish to draw with different a selection highlight color depending on such things as the key state of its *controlView*.

Availability
Available in Mac OS X v10.1 and later.

Declared In
`NSCell.h`

hitTestForEvent:inRect:ofView:

Returns hit testing information for the receiver.

```
- (NSUInteger)hitTestForEvent:(NSEvent *)event inRect:(NSRect)cellFrame
    ofView:(NSView *)controlView
```

Parameters

event
The current event.

cellFrame
The cell's frame.

controlView
The control object in which the cell is located.

Return Value
A constant that specifies the type of area in which the event occurred—see "Hit Testing" (page 620) for values.

Discussion
You can use a bit-wise mask to look for a specific value when calling this method—see "Hit Testing" (page 620) for values.

Generally, this method should be overridden by custom `NSCell` subclasses to return the correct result. Currently, it is called by some multi-cell views, such as `NSTableView`.

By default, `NSCell` looks at the cell type and does the following:

- `NSImageCellType`: If the image exists and the event point is in the image returns `NSCellHitContentArea`, otherwise `NSCellHitNone`.
- `NSTextCellType` (also applies to `NSTextFieldCell`):

  If there is text: If the event point hits in the text, return `NSCellHitContentArea`. Additionally, if the cell is enabled return `NSCellHitContentArea | NSCellHitEditableTextArea`.

  If there is not text: return `NSCellHitNone`.
- `NSNullCellType` (this is the default that applies to non text or image cells who don't override `hitTestForEvent:inRect:ofView:`):

  Return `NSCellHitContentArea` by default;

  If the cell not disabled, and it would track, return `NSCellHitContentArea | NSCellHitTrackableArea`.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
AnimatedTableView
DragNDropOutlineView

Declared In
`NSCell.h`

image

Returns the image displayed by the receiver (if any).

```
- (NSImage *)image
```

Return Value
The image displayed by the receiver, or `nil` if the receiver is not an image-type cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setImage:` (page 589)

Related Sample Code
AnimatedTableView
ComplexBrowser
DragNDropOutlineView

Declared In
NSCell.h

imageRectForBounds:

Returns the rectangle in which the receiver draws its image.

```
- (NSRect)imageRectForBounds:(NSRect)theRect
```

Parameters

theRect

The bounding rectangle of the receiver.

Return Value
The rectangle in which the receiver draws its image. This rectangle is slightly offset from the one in `theRect`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cellSizeForBounds:` (page 549)
- `drawingRectForBounds:` (page 552)

Related Sample Code
AnimatedTableView

Declared In
NSCell.h

importsGraphics

Returns a Boolean value that indicates whether the text of the receiver can contain imported graphics.

```
- (BOOL)importsGraphics
```

Return Value
`YES` if the receiver's text is in the RTFD format and supports imported graphics, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setImportsGraphics:` (page 589)

Declared In
NSCell.h

initImageCell:

Returns an `NSCell` object initialized with the specified image and set to have the cell's default menu.

```
- (id)initImageCell:(NSImage *)anImage
```

Parameters

anImage

The image to use for the cell. If this parameter is `nil`, no image is set.

Return Value

An initialized `NSCell` object, or `nil` if the cell could not be initialized.

Discussion

This is one of four designated initializers you must implement when subclassing. See "Designated Initializers" (page 533) for the complete list.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSCell.h`

initTextCell:

Returns an NSCell object initialized with the specified string and set to have the cell's default menu.

```
- (id)initTextCell:(NSString *)aString
```

Parameters

aString

The initial string to use for the cell.

Return Value

An initialized `NSCell` object, or `nil` if the cell could not be initialized.

Discussion

If no field editor (a shared `NSText` object) has been created for all `NSCell` objects, one is created.

This is one of four designated initializers you must implement when subclassing. See "Designated Initializers" (page 533) for the complete list.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSCell.h`

integerValue

Returns the receiver's value as an `NSInteger`.

```
- (NSInteger)integerValue
```

Return Value

The value of the cell interpreted as an `NSInteger`. If the receiver is not a text-type cell or the cell value is not scannable, returns `0`.

Availability

Available in Mac OS X v10.5 and later.

See Also
- `setIntegerValue:` (page 590)
- `intValue` (page 565)

Declared In
`NSCell.h`

interiorBackgroundStyle

Returns the interior background style for the receiver.

```
- (NSBackgroundStyle)interiorBackgroundStyle
```

Return Value
Returns the interior background style for the receiver.

Discussion
The interior background style describes the surface drawn onto in `drawInteriorWithFrame:inView:` (page 553). This is often the same as the `backgroundStyle` (page 547), but a button that draws a bezel would have a different `interiorBackgroundStyle`.

This is both an override point and a useful method to call. In a custom button with a custom bezel you can override this method to describe that surface. A cell that has custom interior drawing might query this method to help pick an image that looks good on the cell. Calling this method gives you some independence from changes in framework art style.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
PhotoSearch

Declared In
`NSCell.h`

intValue

Returns the receiver's value as an integer.

```
- (int)intValue
```

Return Value
The value of the cell interpreted as an integer. If the receiver is not a text-type cell or the cell value is not scannable, returns `0`.

Discussion
On Mac OS X v10.5 and later, you should use `integerValue` (page 564) instead.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `integerValue` (page 564)

- setIntValue: (page 590)

Related Sample Code
VertexPerformanceTest

Declared In
NSCell.h

isBezeled

Returns a Boolean value that indicates whether the receiver has a bezeled border.

```
- (BOOL)isBezeled
```

Return Value
YES if the receiver has a bezeled border, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setBezeled: (page 581)

Declared In
NSCell.h

isBordered

Returns a Boolean value that indicates whether the receiver has a plain border.

```
- (BOOL)isBordered
```

Return Value
YES if the receiver has a plain border, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setBordered: (page 581)

Declared In
NSCell.h

isContinuous

Returns a Boolean value that indicates whether the receiver's cell sends its action message continuously to its target during mouse tracking.

```
- (BOOL)isContinuous
```

Return Value
YES if the action message should be sent continuously, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setContinuous: (page 582)

Related Sample Code
AnimatedSlider

Declared In
NSCell.h

isEditable

Returns a Boolean value that indicates whether the receiver is editable.

```
- (BOOL)isEditable
```

Return Value
YES if the receiver is editable, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setEditable: (page 584)

Declared In
NSCell.h

isEnabled

Returns a Boolean value that indicates whether the receiver is enabled or disabled.

```
- (BOOL)isEnabled
```

Return Value
YES if the receiver is enabled, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setEnabled: (page 585)

Declared In
NSCell.h

isEntryAcceptable:

Returns whether a string representing a numeric or date value is formatted in a suitable way for the cell's entry type. (Deprecated in Mac OS X v 10.0 and later. Use `NSFormatter` instead.)

```
- (BOOL)isEntryAcceptable:(NSString *)aString
```

Parameters

aString

A string containing the numeric or date value.

Return Value

`YES` if *aString* is formatted appropriately for the receiver, otherwise `NO`.

Availability

Deprecated in Mac OS X v 10.0 and later.

See Also

`- setFormatter:` (page 588)

Declared In

`NSCell.h`

isHighlighted

Returns a Boolean value that indicates whether the receiver is highlighted.

```
- (BOOL)isHighlighted
```

Return Value

`YES` if the receiver has a highlight, otherwise `NO`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSCell.h`

isOpaque

Returns a Boolean value that indicates whether the receiver is opaque (nontransparent).

```
- (BOOL)isOpaque
```

Return Value

`YES` if the receiver is opaque, otherwise `NO` to indicate the receiver might have some transparency.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSCell.h`

isScrollable

Returns a Boolean value that indicates whether the receiver scrolls excess text past the cell's bounds.

```
- (BOOL)isScrollable
```

Return Value
`YES` if excess text scrolls past the cell's bounds, otherwise `NO` (text wrapping is enabled).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setScrollable:` (page 594)

Declared In
`NSCell.h`

isSelectable

Returns a Boolean value that indicates whether the text of the receiver can be selected.

```
- (BOOL)isSelectable
```

Return Value
`YES` if the receiver's text can be selected, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setSelectable:` (page 594)

Declared In
`NSCell.h`

keyEquivalent

Returns the key equivalent to clicking the cell.

```
- (NSString *)keyEquivalent
```

Return Value
An empty string object.

Discussion
Subclasses can override this method to return a string with a valid character for the key equivalent.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSCell.h`

lineBreakMode

Returns the line break mode currently used when drawing text.

```
- (NSLineBreakMode)lineBreakMode
```

Return Value
The line break mode the receiver currently uses when drawing text. See `NSLineBreakMode` (page 1878) for supported values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setLineBreakMode:` (page 591)
- `truncatesLastVisibleLine` (page 610)

Declared In
`NSCell.h`

menu

Returns the receiver's contextual menu.

```
- (NSMenu *)menu
```

Return Value
The receiver's contextual menu, or `nil` if no menu is assigned.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMenu:` (page 591)

Related Sample Code
ClockControl

Declared In
`NSCell.h`

menuForEvent:inRect:ofView:

Returns the menu associated with the receiver and related to the specified event and frame.

```
- (NSMenu *)menuForEvent:(NSEvent *)anEvent inRect:(NSRect)cellFrame ofView:(NSView
    *)aView
```

Parameters
anEvent
The event used to find the menu.

cellFrame
The cell's rectangle. This rectangle indicates the region containing the cursor.

aView
The view that manages the receiver. This is usually the control object that owns the cell.

Return Value
The menu associated with the cell and event parameters, or `nil` if no menu is set.

Discussion
This method is usually invoked by the `NSControl` object (*aView*) managing the receiver. The default implementation simply invokes `menu` (page 570) and returns `nil` if no menu has been set. Subclasses can override to customize the returned menu according to the event received and the area in which the mouse event occurs.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSCell.h`

mnemonic

Returns the character in the receiver's title that appears underlined for use as a mnemonic.

```
- (NSString *)mnemonic
```

Return Value
A string containing the mnemonic character, or an empty string if no mnemonic character is set.

Discussion
Mnemonics are not supported in Mac OS X

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitleWithMnemonic:` (page 599)

Declared In
`NSCell.h`

mnemonicLocation

Returns the position of the underlined mnemonic character in the receiver's title.

```
- (NSUInteger)mnemonicLocation
```

Return Value
A zero-based index into the receiver's title string indicating the position of the character. If there is no mnemonic character, this method returns `NSNotFound`.

Discussion
Mnemonics are not supported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMnemonicLocation:` (page 592)

Declared In
`NSCell.h`

mouseDownFlags

Returns the modifier flags for the last (left) mouse-down event.

```
- (NSInteger)mouseDownFlags
```

Return Value
The modifier flags, or `0` if tracking has not yet occurred or no modifier keys accompanied the mouse-down event.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `modifierFlags` (page 1082) (`NSEvent`)

Declared In
`NSCell.h`

nextState

Returns the receiver's next state.

```
- (NSInteger)nextState
```

Return Value
The receiver's next state (for possible values, see "`NSCellStateValue`" (page 618)).

Discussion
If the receiver has three states, it cycles through them in this order: on, off, mixed, on, and so forth. If the receiver has two states, it toggles between them.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsMixedState` (page 545)
- `setAllowsMixedState:` (page 578)
- `setNextState` (page 592)

Declared In
`NSCell.h`

objectValue

Returns the receiver's value as an Objective-C object

```
- (id)objectValue
```

Return Value
The receiver's object value, or `nil` if a valid object has not been associated with the receiver.

Discussion
To be valid object value, the receiver must have a formatter capable of converting the object to and from its textual representation.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setObjectValue:` (page 592)

Related Sample Code
ClockControl
PhotoSearch

Declared In
`NSCell.h`

performClick:

Simulates a single mouse click on the receiver.

```
- (void)performClick:(id)sender
```

Parameters

sender

The object to use as the sender of the event (if the receiver's control view is not valid). This object must be a subclass of `NSView`.

Discussion
This method performs the receiver's action on its target. The receiver must be enabled to perform the action. If the receiver's control view is valid, that view is used as the sender; otherwise, the value in *sender* is used.

The receiver of this message must be a cell of type `NSActionCell`. This method raises an exception if the action message cannot be successfully sent.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- controlView` (page 552)

Declared In
`NSCell.h`

refusesFirstResponder

Returns a Boolean value that indicates whether the receiver should not become the first responder.

```
- (BOOL)refusesFirstResponder
```

Return Value
`YES` if the receiver should never become the first responder, otherwise `NO` if the receiver can become the first responder.

Discussion
To find out whether the receiver can become first responder at this time, use the method `acceptsFirstResponder` (page 544).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setRefusesFirstResponder:` (page 593)

Declared In
`NSCell.h`

representedObject

Returns the object the receiver represents.

```
- (id)representedObject
```

Return Value
The object represented by the receiver.

Discussion
Represented objects let you link a cell to an appropriate object. For example, you could have a pop-up list of color names, and the represented objects could be the appropriate `NSColor` objects.

Special Considerations

Note that if you copy an `NSCell` instance, the represented object in the copy is set to `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setRepresentedObject:` (page 593)

Related Sample Code
NewsReader

Declared In
`NSCell.h`

resetCursorRect:inView:

Sets the receiver to show the I-beam cursor while it tracks the mouse.

```
- (void)resetCursorRect:(NSRect)cellFrame inView:(NSView *)controlView
```

Parameters

cellFrame

The rectangle in which to display the I-beam cursor.

controlView

The control that manages the cell.

Discussion

The receiver must be an enabled and selectable (or editable) text-type cell.

This method is invoked by `resetCursorRects` (page 3206) and in general you do not need to call this method unless you have a custom `NSView` that uses a cell.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSCell.h`

selectWithFrame:inView:editor:delegate:start:length:

Selects the specified text range in the cell's field editor.

```
- (void)selectWithFrame:(NSRect)aRect inView:(NSView *)controlView editor:(NSText
    *)textObj delegate:(id)anObject start:(NSInteger)selStart
  length:(NSInteger)selLength
```

Parameters

aRect

The bounding rectangle of the cell.

controlView

The control that manages the cell.

textObj

The field editor to use for editing the cell.

anObject

The object to use as a delegate for the field editor (`textObj` parameter). This delegate object receives various `NSText` delegation and notification methods during the course of editing the cell's contents.

selStart

The start of the text selection.

selLength

The length of the text range.

Discussion

This method is similar to `editWithFrame:inView:editor:delegate:event:` (page 555), except that it can be invoked in any situation, not only on a mouse-down event. This method returns without doing anything if *controlView*, *textObj*, or the receiver is `nil`, or if the receiver has no font set for it.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSCell.h`

sendActionOn:

Sets the conditions on which the receiver sends action messages to its target.

```
- (NSInteger)sendActionOn:(NSInteger)mask
```

Parameters

mask

A bit mask containing the conditions for sending the action. The only conditions that are actually checked are associated with the `NSLeftMouseDownMask` (page 1097), `NSLeftMouseUpMask` (page 1097), `NSLeftMouseDraggedMask` (page 1098), and `NSPeriodicMask` (page 1099) bits.

Return Value
A bit mask containing the previous settings. This bit mask uses the same values as specified in the *mask* parameter.

Discussion
You use this method during mouse tracking when the mouse button changes state, the mouse moves, or if the cell is marked to send its action continuously while tracking. Because of this, the only bits checked in *mask* are `NSLeftMouseDownMask` (page 1097), `NSLeftMouseUpMask` (page 1097), `NSLeftMouseDraggedMask` (page 1098), and `NSPeriodicMask` (page 1099), which are declared in the `NSEvent` class reference.

You can use the `setContinuous:` (page 582) method to turn on the flag corresponding to `NSPeriodicMask` (page 1099) or `NSLeftMouseDraggedMask` (page 1098), whichever is appropriate to the given subclass of `NSCell`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- action` (page 544)

Declared In
`NSCell.h`

sendsActionOnEndEditing

Returns a Boolean value that indicates whether the receiver's `NSControl` object sends its action message whenever the user finishes editing the cell's text.

```
- (BOOL)sendsActionOnEndEditing
```

Return Value
`YES` if the receiver's control sends its action message when editing is complete, otherwise `NO`.

Discussion
If this method returns `YES`, the receiver's `NSControl` object sends its action message when the user does one of the following:

- Presses the Return key
- Presses the Tab key to move out of the field
- Clicks another text field

If it returns NO, the cell's NSControl object sends its action message only when the user presses the Return key.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setSendsActionOnEndEditing: (page 595)

Declared In
NSCell.h

setAction:

Sets the cell's action method to the specified selector.

```
- (void)setAction:(SEL)aSelector
```

Parameters

aSelector

The new action-message selector to associate with the receiver's cell. Specify NULL to prevent action messages from being sent to the receiver's target.

Discussion
The NSCell implementation of this method raises NSInternalInconsistencyException. Subclasses (such as NSActionCell) override this method to set the action method as part of the target/action implementation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- action (page 544)
- setTarget: (page 598)
- target (page 608)

Declared In
NSCell.h

setAlignment:

Sets the alignment of text in the receiver.

```
- (void)setAlignment:(NSTextAlignment)mode
```

Parameters

mode

This value can be one of the following constants: NSLeftTextAlignment, NSRightTextAlignment,NSCenterTextAlignment, NSJustifiedTextAlignment, orNSNaturalTextAlignment.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alignment` (page 545)
- `setWraps:` (page 601)

Related Sample Code
FunHouse

Declared In
`NSCell.h`

setAllowsEditingTextAttributes:

Sets whether the receiver allows the user to edit textual attributes of its contents.

```
- (void)setAllowsEditingTextAttributes:(BOOL)flag
```

Parameters

flag

If `YES`, the user can modify the font and other textual attributes of the cell's text. If `NO`, the user cannot edit the text or import graphics, which effectively means the cell cannot support RTFD text.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsEditingTextAttributes` (page 545)
- `setImportsGraphics:` (page 589)

Declared In
`NSCell.h`

setAllowsMixedState:

Sets whether the receiver supports three states or just two.

```
- (void)setAllowsMixedState:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver supports three states (on, off, and mixed); otherwise, if `NO`, the receiver supports only two states (on and off).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsMixedState` (page 545)
- `nextState` (page 572)
- `setNextState` (page 592)

Declared In
`NSCell.h`

setAllowsUndo:

Sets whether the receiver assumes responsibility for undo operations within the cell.

```
- (void)setAllowsUndo:(BOOL)allowsUndo
```

Parameters

allowsUndo

If YES, the receiver handles undo operations; otherwise, if NO, the application's custom undo manager handles undo operations.

Discussion

Subclasses invoke this method to indicate their preference for handling undo operations; otherwise, you should not need to call this method directly.

Availability

Available in Mac OS X v10.4 and later.

See Also

- allowsUndo (page 546)

Declared In

NSCell.h

setAttributedStringValue:

Sets the value of the receiver's cell using an attributed string.

```
- (void)setAttributedStringValue:(NSAttributedString *)attribStr
```

Parameters

attribStr

The value of the cell interpreted as an attributed string.

Discussion

If a formatter is set for the receiver, but the formatter does not understand the attributed string, it marks *attribStr* as an invalid object. If the receiver is not a text-type cell, it is converted to one before the value is set.

For Mac OS X v10.3 and later: If you use a class that responds to the selector attributedStringValue (page 546) for the object value of a cell, then the cell uses that method to fetch the string to draw rather than using stringValue (page 605).

The following example sets the text in a cell to 14 points, red, in the system font.

```
NSColor *txtColor = [NSColor redColor];
NSFont *txtFont = [NSFont boldSystemFontOfSize:14];
NSDictionary *txtDict = [NSDictionary dictionaryWithObjectsAndKeys:
        txtFont, NSFontAttributeName, txtColor, NSForegroundColorAttributeName,
 nil];
NSAttributedString *attrStr = [[[NSAttributedString alloc]
         initWithString:@"Hello!" attributes:txtDict] autorelease];
[[attrStrTextField cell] setAttributedStringValue:attrStr];
[attrStrTextField updateCell:[attrStrTextField cell]];
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- attributedStringValue (page 546)

Declared In
NSCell.h

setBackgroundStyle:

Sets the background style for the receiver.

```
- (void)setBackgroundStyle:(NSBackgroundStyle)style
```

Parameters

style
The background style for the receiver.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
TrackBall

Declared In
NSCell.h

setBaseWritingDirection:

Sets the initial writing direction used to determine the actual writing direction for text .

```
- (void)setBaseWritingDirection:(NSWritingDirection)writingDirection
```

Parameters

writingDirection
The initial writing direction the receiver uses to determine the actual writing direction for text. See NSWritingDirection (page 2748) for possible values.

Discussion
If you know the base writing direction of the text you are rendering, you can use this method to specify that direction to the text system.

Availability
Available in Mac OS X v10.4 and later.

See Also
- baseWritingDirection (page 547)

Declared In
NSCell.h

setBezeled:

Sets whether the receiver draws itself with a bezeled border.

```
- (void)setBezeled:(BOOL)flag
```

Parameters

flag

If YES, the receiver uses a bezeled border.

Discussion

The setBezeled: and setBordered: (page 581) methods are mutually exclusive (that is, a border can be only plain or bezeled). Invoking this method automatically removes any border that had already been set, regardless of the value in the flag parameter.

Availability

Available in Mac OS X v10.0 and later.

See Also

- isBezeled (page 566)

Related Sample Code

QTKitMovieShuffler

Declared In

NSCell.h

setBordered:

Sets whether the receiver draws itself outlined with a plain border.

```
- (void)setBordered:(BOOL)flag
```

Parameters

flag

If YES, the receiver uses a plain border.

Discussion

The setBezeled: (page 581) and setBordered: methods are mutually exclusive (that is, a border can be only plain or bezeled). Invoking this method automatically removes any bezel that had already been set, regardless of the value in the flag parameter.

Availability

Available in Mac OS X v10.0 and later.

See Also

- isBordered (page 566)

Declared In

NSCell.h

setCellAttribute:to:

Sets the value for the specified cell attribute.

```
- (void)setCellAttribute:(NSCellAttribute)aParameter to:(NSInteger)value
```

Parameters

aParameter

The cell attribute whose value you want to set. Attributes include the receiver's current state and whether it is disabled, editable, or highlighted.

value

The new value for the attribute.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cellAttribute:` (page 548)

Declared In
`NSCell.h`

setContinuous:

Sets whether the receiver's cell sends its action message continuously to its target during mouse tracking.

```
- (void)setContinuous:(BOOL)flag
```

Parameters

flag

If `YES`, the action message should be sent continuously.

Discussion
In practice, the continuous setting of action messages has meaning only for `NSActionCell` and its subclasses, which implement the target/action mechanism. Some `NSControl` subclasses, notably `NSMatrix`, send a default action to a default target when a cell doesn't provide a target or action.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isContinuous` (page 566)
- `sendActionOn:` (page 576)

Declared In
`NSCell.h`

setControlSize:

Sets the size of the receiver.

```
- (void)setControlSize:(NSControlSize)size
```

Parameters

size

A value that specifies the size of the receiver (for possible values, see "`NSControlSize`" (page 620)).

Discussion
Changing the cell's control size does not change the font of the cell. Use the `systemFontSizeForControlSize:` (page 1159) class method of `NSFont` to obtain the system font based on the new control size and set it.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- controlSize` (page 551)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSCell.h`

setControlTint:

Sets the receiver's control tint.

```
- (void)setControlTint:(NSControlTint)controlTint
```

Parameters
controlTint
An `NSControlTint` (page 533) value that specifies the tint of the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- controlTint` (page 551)

Declared In
`NSCell.h`

setControlView:

Sets the receiver's control view.

```
- (void)setControlView:(NSView *)view
```

Parameters
view
The view (normally an `NSControl` object) to associate with the cell.

Discussion
The control view represents the control currently being rendered by the cell.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `controlView` (page 552)

Related Sample Code
PhotoSearch

Declared In
`NSCell.h`

setDoubleValue:

Sets the value of the receiver's cell using a double-precision floating-point number.

```
- (void)setDoubleValue:(double)aDouble
```

Parameters

aDouble

The value of the cell interpreted as a double-precision floating-point number.

Discussion
In its implementation, this method invokes the `setObjectValue:` (page 592) method to set the actual value. This method does nothing if the receiver is not a text-type cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `doubleValue` (page 552)

Declared In
`NSCell.h`

setEditable:

Sets whether the user can edit the receiver's text.

```
- (void)setEditable:(BOOL)flag
```

Parameters

flag

If `YES`, the user is allowed to edit the receiver's text. If this value is `YES`, the text is also made selectable. If it is `NO`, the selectable attribute is restored to the value it was before the cell was last made editable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isEditable` (page 567)
- `setSelectable:` (page 594)

Related Sample Code
ImageBackground
QTAudioContextInsert

QTAudioExtractionPanel
QTKitMovieShuffler
SourceView

Declared In
NSCell.h

setEnabled:

Sets whether the receiver is enabled or disabled.

```
- (void)setEnabled:(BOOL)flag
```

Parameters

flag

If YES the receiver is enabled; otherwise, if NO, the receiver is disabled.

Discussion
The text of disabled cells is changed to gray. If a cell is disabled, it cannot be highlighted, does not support mouse tracking (and thus cannot participate in target/action functionality), and cannot be edited. However, you can still alter many attributes of a disabled cell programmatically. (The setState: (page 596) method, for instance, still works.)

Availability
Available in Mac OS X v10.0 and later.

See Also
- isEnabled (page 567)

Related Sample Code
DragNDropOutlineView
QTKitMovieShuffler
TextLinks
VertexPerformanceTest

Declared In
NSCell.h

setEntryType:

Sets how numeric data is formatted in the receiver and places restrictions on acceptable input. (Deprecated in Mac OS X v10.0 and later. Use a formatter instead—see setFormatter: (page 588).)

```
- (void)setEntryType:(NSInteger)aType
```

Parameters

aType

One of the types listed for this method in "Data Entry Types" (page 622).

Availability
Deprecated in Mac OS X v10.0 and later.

See Also
- `setFormatter:` (page 588)

Declared In
`NSCell.h`

setFloatingPointFormat:left:right:

Sets the auto-ranging and floating point number format of the receiver's cell. (Deprecated in Mac OS X v10.0. Use a formatter instead. See `setFormatter:` (page 588))

```
- (void)setFloatingPointFormat:(BOOL)autoRange left:(NSUInteger)leftDigits
    right:(NSUInteger)rightDigits
```

Parameters

autoRange

If `YES`, auto-ranging is enabled, otherwise it is disabled.

leftDigits

The number of digits to display to the left of the decimal point.

rightDigits

The number of digits to display to the right of the decimal point.

Discussion

Sets whether floating-point numbers are auto-ranged in the receiver and sets the sizes of the fields to the left and right of the decimal point. If *autoRange* is `NO`, *leftDigits* specifies the maximum number of digits to the left of the decimal point, and *rightDigits* specifies the number of digits to the right (the fractional digit places will be padded with zeros to fill this width). However, if a number is too large to fit its integer part in *leftDigits* digits, as many places as are needed on the left are effectively removed from *rightDigits* when the number is displayed.

If *autoRange* is `YES`, *leftDigits* and *rightDigits* are simply added to form a maximum total field width for the receiver (plus 1 for the decimal point). The fractional part will be padded with zeros on the right to fill this width, or truncated as much as possible (up to removing the decimal point and displaying the number as an integer). The integer portion of a number is never truncated—that is, it is displayed in full no matter what the field width limit is.

The following example sets a cell used to display dollar amounts up to 99,999.99:

```
[[currencyDollarsField cell] setEntryType:NSFloatType];
[[currencyDollarsField cell] setFloatingPointFormat:NO left:5  right:2];
```

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.0.

See Also
- `setFormatter:` (page 588)

Declared In
`NSCell.h`

setFloatValue:

Sets the value of the receiver's cell using a single-precision floating-point number.

```
- (void)setFloatValue:(float)aFloat
```

Parameters

aFloat

The value of the cell interpreted as a single-precision floating-point number.

Discussion

In its implementation, this method invokes the `setObjectValue:` (page 592) method to set the actual value. This method does nothing if the receiver is not a text-type cell.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- floatValue` (page 557)

Declared In

`NSCell.h`

setFocusRingType:

Sets the type of focus ring to be used.

```
- (void)setFocusRingType:(NSFocusRingType)focusRingType
```

Parameters

focusRingType

Possible values are listed in `NSFocusRingType` (page 4009). To disable a view's focus ring, specify `NSFocusRingTypeNone`.

Discussion

You should only disable a view from drawing its focus ring if you want to draw your own focus ring, or if there is not sufficient space to display a focus ring in the default location.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- focusRingType` (page 558)

`+ defaultFocusRingType` (page 543)

Declared In

`NSCell.h`

setFont:

Sets the font to use when the receiver displays text.

```
- (void)setFont:(NSFont *)fontObj
```

Parameters

fontObj

The font to use.

Discussion

If the receiver is not a text-type cell, the method converts it to that type before setting the font.

Availability

Available in Mac OS X v10.0 and later.

See Also

- font (page 558)

Related Sample Code

FunHouse

Mountains

Declared In

NSCell.h

setFormatter:

Sets the receiver's formatter object.

```
- (void)setFormatter:(NSFormatter *)newFormatter
```

Parameters

newFormatter

The formatter to use with the cell, or nil if you do not want the cell to use a formatter.

Discussion

Cells use a formatter object to format the textual representation of their object value and to validate cell input and convert that input to an object value. If the new formatter cannot interpret the receiver's current object value, that value is converted to a string object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- formatter (page 559)

Declared In

NSCell.h

setHighlighted:

Sets whether the receiver has a highlighted appearance.

```
- (void)setHighlighted:(BOOL)flag
```

Parameters

flag

If YES, the receiver has a highlight.

Discussion
By default, this method does nothing. The `NSButtonCell` class overrides this method to draw the button with the appearance specified by `NSCellLightsByBackground`, `NSCellLightsByContents`, or `NSCellLightsByGray`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSCell.h`

setImage:

Sets the image to be displayed by the receiver.

```
- (void)setImage:(NSImage *)image
```

Parameters

image
The image to display in the cell.

Discussion
If the receiver is not an image-type cell, the method converts it to that type of cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `image` (page 562)
- `setType:` (page 599)

Related Sample Code
DragNDropOutlineView
FunkyOverlayWindow
QTKitMovieShuffler
SourceView
Transformed Image

Declared In
`NSCell.h`

setImportsGraphics:

Sets whether the receiver can import images into its text.

```
- (void)setImportsGraphics:(BOOL)flag
```

Parameters

flag
If `YES`, the receiver can import images into its text and support RTFD text. If `NO`, RTFD text is not supported.

Discussion
If *flag* is YES, the receiver is also set to allow editing of text attributes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- importsGraphics (page 563)
- setAllowsEditingTextAttributes: (page 578)

Declared In
NSCell.h

setIntegerValue:

Sets the value of the receiver using an NSInteger.

```
- (void)setIntegerValue:(NSInteger)anInteger
```

Parameters
anInteger
The value of the cell interpreted as an NSInteger.

Discussion
In its implementation, this method invokes the setObjectValue: (page 592) method to set the actual value. This method does nothing if the receiver is not a text-type cell.

Availability
Available in Mac OS X v10.5 and later.

See Also
- integerValue (page 564)
- setIntValue: (page 590)

Declared In
NSCell.h

setIntValue:

Sets the value of the receiver using an integer.

```
- (void)setIntValue:(int)anInt
```

Parameters
anInt
The value of the cell interpreted as an integer.

Discussion
In its implementation, this method invokes the setObjectValue: (page 592) method to set the actual value. This method does nothing if the receiver is not a text-type cell.

On Mac OS X v10.5 and later, you should use setIntegerValue: (page 590) instead.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `intValue` (page 565)
- `setIntegerValue:` (page 590)

Declared In
`NSCell.h`

setLineBreakMode:

Sets the line break mode to use when drawing text

```
- (void)setLineBreakMode:(NSLineBreakMode)mode
```

Parameters

mode

The line break mode the receiver currently uses when drawing text. See `NSLineBreakMode` (page 1878) for supported values.

Discussion
The line break mode can also be modified by calling the `setWraps:` (page 601) method.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `lineBreakMode` (page 570)
- `setWraps:` (page 601)

Declared In
`NSCell.h`

setMenu:

Sets the contextual menu for the cell.

```
- (void)setMenu:(NSMenu *)aMenu
```

Parameters

aMenu

A menu that has commands contextually related to the receiver. Specify `nil` to clear the previous menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `menu` (page 570)

Related Sample Code
DragNDropOutlineView

Declared In
NSCell.h

setMnemonicLocation:

Sets the character of the receiver's title to be used as a mnemonic character.

```
- (void)setMnemonicLocation:(NSUInteger)location
```

Parameters

location

The zero-based index into the cell's title string specifying the location of the mnemonic character. The specified character is underlined when the title is drawn.

Discussion
Mnemonics are not supported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

See Also
- mnemonicLocation (page 571)

Declared In
NSCell.h

setNextState

Changes the state of the receiver to its next state.

```
- (void)setNextState
```

Discussion
If the receiver has three states, it cycles through them in this order: on, off, mixed, on, and so forth. If the receiver has two states, it toggles between them.

Availability
Available in Mac OS X v10.0 and later.

See Also
- allowsMixedState (page 545)
- nextState (page 572)
- setAllowsMixedState: (page 578)

Declared In
NSCell.h

setObjectValue:

Sets the receiver's object value.

```
- (void)setObjectValue:(id < NSCopying >)object
```

Parameters

object

The new object value for the cell.

Discussion

To be valid object value, the receiver must have a formatter capable of converting the object to and from its textual representation.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `objectValue` (page 572)
- `setRepresentedObject:` (page 593)

Related Sample Code

STUCAuthoringDeviceCocoaSample

Declared In

`NSCell.h`

setRefusesFirstResponder:

Sets whether the receiver should not become the first responder.

```
- (void)setRefusesFirstResponder:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver should never become the first responder; otherwise, it may become the first responder.

Discussion

If `refusesFirstResponder` (page 573) returns `NO` and the cell is enabled, the method `acceptsFirstResponder` (page 544) returns `YES`, allowing the cell to become first responder.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSCell.h`

setRepresentedObject:

Sets the object represented by the receiver.

```
- (void)setRepresentedObject:(id)anObject
```

Parameters

anObject

The object to associate with the receiver.

Discussion
You can use this method to link two objects together. For example, if the receiver's title was "Blue", you could associate an `NSColor` object whose color was set to blue.

Special Considerations

Note that if you copy an `NSCell` instance, the represented object in the copy is set to `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setObjectValue:` (page 592)
- `representedObject` (page 574)

Related Sample Code
NewsReader

Declared In
`NSCell.h`

setScrollable:

Sets whether excess text in the receiver is scrolled past the cell's bounds.

```
- (void)setScrollable:(BOOL)flag
```

Parameters

flag

If `YES`, text can be scrolled past the cell's bounds; otherwise, if `NO`, the text wrapping is enabled.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isScrollable` (page 569)

Declared In
`NSCell.h`

setSelectable:

Sets whether text in the receiver can be selected.

```
- (void)setSelectable:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver's text can be selected. If this value is `NO`, editability is also disabled; if it is `YES`, editability is not affected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isSelectable` (page 569)
- `setEditable:` (page 584)

Declared In
`NSCell.h`

setSendsActionOnEndEditing:

Sets whether the receiver's NSControl object sends its action message whenever the user finishes editing the cell's text.

```
- (void)setSendsActionOnEndEditing:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver's control sends its action message when editing is complete; otherwise, if `NO`, it sends the action message only when the user presses the Return key.

Discussion
If *flag* is `YES`, the receiver's `NSControl` object sends its action message when the user does one of the following:

- Presses the Return key
- Presses the Tab key to move out of the field
- Clicks another text field

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sendsActionOnEndEditing` (page 576)

Related Sample Code
Quartz Composer QCTV

Declared In
`NSCell.h`

setShowsFirstResponder:

Sets whether the receiver draws some indication of its first responder status.

```
- (void)setShowsFirstResponder:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver draws an indication of its first responder status, otherwise it does not.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `showsFirstResponder` (page 602)

Declared In
`NSCell.h`

setState:

Sets the receiver's state to the specified value.

```
- (void)setState:(NSInteger)value
```

Parameters

value

The possible state values are `NSOnState`, `NSOffState`, and `NSMixedState`. If the cell supports only two states and you specify `NSMixedState`, this method sets the state to `NSOnState`.

Discussion
The `NSOffState` state indicates the normal or unpressed state. The `NSOnState` state indicates the alternate or pressed state. The NSMixedState state indicates that the feature represented by the control is in effect somewhere.

Although using the enumerated constants is preferred, *value* can also be an integer. If the cell has two states, 0 is treated as `NSOffState`, and a nonzero value is treated as `NSOnState`. If the cell has three states, 0 is treated as `NSOffState`; a negative value, as `NSMixedState`; and a positive value, as `NSOnState`.

Note that the value `state` (page 603) returns may not be the same value you passed into the `value` parameter.

To check whether the cell has three states (and uses the mixed state), invoke the `allowsMixedState` (page 545) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `state` (page 603)
- `setAllowsMixedState:` (page 578)

Related Sample Code
EnhancedDataBurn

Declared In
`NSCell.h`

setStringValue:

Sets the value of the receiver's cell using an `NSString` object.

```
- (void)setStringValue:(NSString *)aString
```

Parameters

aString

The string value of the cell.

Discussion
In its implementation, this method invokes the `setObjectValue:` (page 592) method to set the actual value. If no formatter is assigned to the receiver or if the formatter cannot "translate" *aString* to an underlying object, the receiver is flagged as having an invalid object. If the receiver is not a text-type cell, this method converts it to one before setting the object value.

For Mac OS X v10.3 and later: If you use a class that responds to the selector `attributedStringValue` (page 546) for the object value of a cell, the cell uses that method to fetch the string to draw rather than the `stringValue` (page 605) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `stringValue` (page 605)

Related Sample Code
DragNDropOutlineView
NewsReader
QTKitMovieShuffler
SimpleComboBox

Declared In
`NSCell.h`

setTag:

Sets the tag of the receiver.

```
- (void)setTag:(NSInteger)anInteger
```

Parameters
anInteger
The new tag for the cell.

Discussion
The `NSCell` implementation of this method raises `NSInternalInconsistencyException`. The `NSActionCell` implementation sets the receiver's tag integer to *anInteger*.

Tags allow you to identify particular cells. Tag values are not used internally; they are only changed by external invocations of `setTag:`. You typically set tag values in Interface Builder and use them at runtime in your application. When you set the tag of a control with a single cell in Interface Builder, it sets the tags of both the control and the cell to the same value as a convenience.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `tag` (page 605)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSCell.h

setTarget:

Sets the target object to receive action messages.

```
- (void)setTarget:(id)anObject
```

Parameters

anObject
The new target object to associate with the receiver's cell, or `nil` to remove the current target.

Discussion
The `NSCell` implementation of this method raises `NSInternalInconsistencyException`. Subclasses (such as `NSActionCell`) override this method to set the target object as part of the target/action implementation.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- target` (page 608)

Declared In
NSCell.h

setTitle:

Sets the title of the receiver.

```
- (void)setTitle:(NSString *)aString
```

Parameters

aString
The new string value for the cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- title` (page 609)

Related Sample Code
ButtonMadness
ObjectPath
Quartz Composer WWDC 2005 TextEdit
SimpleCocoaBrowser
TextLinks

Declared In
NSCell.h

setTitleWithMnemonic:

Sets the title of the receiver with one character in the string denoted as an access key.

```
- (void)setTitleWithMnemonic:(NSString *)aString
```

Parameters

aString

The new title of the cell. One character in the string should be preceded by an ampersand (&) character. The character that follows becomes the mnemonic character for the title.

Discussion

Mnemonics are not supported in Mac OS X.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `mnemonic` (page 571)
- `setMnemonicLocation:` (page 592)

Declared In

`NSCell.h`

setTruncatesLastVisibleLine:

Sets whether the receiver truncates and adds the ellipsis character to the last visible line if the text doesn't fit into the cell bounds.

```
- (void)setTruncatesLastVisibleLine:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver truncates the last line; if `NO`, it does not truncate.

Discussion

The line break mode must be either `NSLineBreakByWordWrapping` or `NSLineBreakByCharWrapping`. Otherwise, this setting is ignored.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `truncatesLastVisibleLine` (page 610)
- `setLineBreakMode:` (page 591)

Declared In

`NSCell.h`

setType:

Sets the type of the cell, changing it to a text cell, image cell, or null cell.

```
- (void)setType:(NSCellType)aType
```

Parameters

aType

The new type of the cell (see "NSCellType" (page 613) for possible values).

Discussion

If the cell is already the same type as the one specified in the *aType* parameter, this method does nothing.

If *aType* is NSTextCellType, this method converts the receiver to a cell of that type, giving it a default title and setting the font to the system font at the default size. If *aType* is NSImageCellType, the cell type is not changed until you set a new non-nil image.

Availability

Available in Mac OS X v10.0 and later.

See Also

- type (page 611)
- setImage: (page 589)

Declared In

NSCell.h

setUpFieldEditorAttributes:

Configures the textual and background attributes of the receiver's field editor.

```
- (NSText *)setUpFieldEditorAttributes:(NSText *)textObj
```

Parameters

textObj

The field editor to configure. .

Return Value

The configured field editor.

Discussion

If the receiver is disabled, this method sets the text color to dark gray; otherwise the method sets it to the default color. If the receiver has a bezeled border, this method sets the background to the default color for text backgrounds; otherwise, the method sets it to the color of the receiver's NSControl object.

You should not use this method to substitute a new field editor. setUpFieldEditorAttributes: (page 600) is intended to modify the attributes of the text object (that is, the field editor) passed into it and return that text object. If you want to substitute your own field editor, use the fieldEditor:forObject: (page 3325) method or the windowWillReturnFieldEditor:toObject: delegate method of NSWindow.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSCell.h

setUserInterfaceLayoutDirection:

Sets the layout direction of the user interface.

```
- (void)setUserInterfaceLayoutDirection:(NSUserInterfaceLayoutDirection)layoutDirection
```

Parameters

layoutDirection

The direction of the user interface layout. See `NSUserInterfaceLayoutDirection` (page 184) for possible values.

Discussion

This method specifies the general user interface layout flow directions. For `NSCell` subclasses that have multiple visual components in a single cell instance, this property should specify the directionality or flow of components. It affects, for example, the layout of an `NSForm` object's title and value fields, the position of a disclosure triangle, and so on.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- userInterfaceLayoutDirection` (page 611)

Declared In

`NSCell.h`

setUsesSingleLineMode:

Sets whether the text cell restricts layout and rendering of its content to a single line.

```
- (void)setUsesSingleLineMode:(BOOL)flag
```

Parameters

flag

`YES` if layout and rendering should be restricted to a single line, otherwise `NO`.

Discussion

If YES, the cell ignores the return value from `wraps` (page 612), interprets `NSLineBreakByWordWrapping` and `NSLineBreakByCharWrapping` returned by `lineBreakMode` (page 570) as `NSLineBreakByClipping`, and configures the field editor to ignore key binding commands that insert paragraph and line separators.

The field editor bound to a single line cell filters paragraph and line separator insertion from user actions. Cells in the single line mode use the fixed baseline layout. The text baseline position is determined solely by the control size regardless of content font style or size.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- usesSingleLineMode` (page 612)

Declared In

`NSCell.h`

setWraps:

Sets whether text in the receiver wraps when its length exceeds the frame of the cell.

```
- (void)setWraps:(BOOL)flag
```

Parameters

flag

If YES, the receiver wraps text and also makes the receiver non-scrollable; otherwise, if NO, text is not wrapped.

Discussion

If the text of the receiver is an attributed string value you must explicitly set the paragraph style line break mode. Calling this method with the value YES is equivalent to calling the setLineBreakMode: method with the value NSLineBreakByWordWrapping.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setLineBreakMode: (page 591)
- wraps (page 612)

Related Sample Code

Quartz Composer QCTV

Declared In

NSCell.h

showsFirstResponder

Returns a Boolean value that indicates whether the receiver should draw some indication of its first responder status.

```
- (BOOL)showsFirstResponder
```

Return Value

YES if the receiver should draw an indication of its first responder status, otherwise NO.

Discussion

The NSCell class itself does not draw a first-responder indicator. Subclasses may use the returned value to determine whether or not they should draw one, however.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setShowsFirstResponder: (page 595)

Related Sample Code

ClockControl

Declared In

NSCell.h

startTrackingAt:inView:

Begins tracking mouse events within the receiver.

```
- (BOOL)startTrackingAt:(NSPoint)startPoint inView:(NSView *)controlView
```

Parameters

startPoint

The initial location of the cursor.

controlView

The `NSControl` object managing the receiver.

Return Value

`YES` if the receiver is set to respond continuously or set to respond when the mouse is dragged, otherwise `NO`.

Discussion

The `NSCell` implementation of `trackMouse:inRect:ofView:untilMouseUp:` (page 610) invokes this method when tracking begins. Subclasses can override this method to implement special mouse-tracking behavior at the beginning of mouse tracking—for example, displaying a special cursor.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `continueTracking:at:inView:` (page 550)
- `stopTracking:at:inView:mouseIsUp:` (page 604)

Related Sample Code

QTKitMovieShuffler

Declared In

`NSCell.h`

state

Returns the receiver's state.

```
- (NSInteger)state
```

Return Value

The receiver's state (for possible values, see "`NSCellStateValue`" (page 618)).

Discussion

Cells can have two or three states. If the receiver has two states, it returns either `NSOffState` (the normal or unpressed state) or `NSOnState` (the alternate or pressed state). If it has three, it may also return `NSMixedState`, indicating the feature is in effect somewhere.

To check whether the receiver uses the mixed state, use the method `allowsMixedState` (page 545).

Note that the value `state` (page 603) returns may not be the same value you passed into `setState:` (page 596).

Availability

Available in Mac OS X v10.0 and later.

See Also
- `setState:` (page 596)
- `setAllowsMixedState:` (page 578)

Declared In
`NSCell.h`

stopTracking:at:inView:mouseIsUp:

Stops tracking mouse events within the receiver.

```
- (void)stopTracking:(NSPoint)lastPoint at:(NSPoint)stopPoint inView:(NSView
    *)controlView mouseIsUp:(BOOL)flag
```

Parameters

lastPoint
Contains the previous position of the cursor.

stopPoint
The current location of the cursor.

controlView
The `NSControl` object managing the receiver.

flag
If `YES`, this method was invoked because the user released the mouse button; otherwise, if `NO`, the cursor left the designated tracking rectangle.

Discussion
The default `NSCell` implementation of `trackMouse:inRect:ofView:untilMouseUp:` (page 610) invokes this method when the cursor has left the bounds of the receiver or the mouse button goes up. The default `NSCell` implementation of this method does nothing. Subclasses often override this method to provide customized tracking behavior. The following example increments the state of a tristate cell when the mouse button is clicked:

```
- (void)stopTracking:(NSPoint)lastPoint at:(NSPoint)stopPoint
    inView:(NSView *)controlView mouseIsUp:(BOOL)flag
{
    if (flag == YES) {
        [self setTriState:([self triState]+1)];
    }
}
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `startTrackingAt:inView:` (page 603)
- `continueTracking:at:inView:` (page 550)

Related Sample Code
QTKitMovieShuffler

Declared In
`NSCell.h`

stringValue

Returns the value of the receiver's cell as an `NSString` object.

```
- (NSString *)stringValue
```

Return Value
The string value of the cell. This value may be an interpreted version of the cell's actual value. Interpretations are performed by the cell's formatter.

Discussion
If no formatter exists and the cell's value is an `NSString` object, this method returns the value as a plain, attributed, or localized formatted string. If the value is not an `NSString` object or cannot be converted to one, this method returns an empty string.

For Mac OS X v10.3 and later: If you use a class that responds to the selector `attributedStringValue` (page 546) for the object value of a cell, the cell uses that method to fetch the string to draw rather than the `stringValue` method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setStringValue:` (page 596)

Related Sample Code
QTKitMovieShuffler
SimpleComboBox

Declared In
`NSCell.h`

tag

Returns the tag identifying the receiver.

```
- (NSInteger)tag
```

Return Value
The tag value. The `NSCell` implementation of this method returns `-1`.

Discussion
Tags allow you to identify particular cells. Tag values are not used internally; they are only changed by external invocations of `setTag:`. You typically set tag values in Interface Builder and use them at runtime in your application. When you set the tag of a control with a single cell in Interface Builder, it sets the tags of both the control and the cell to the same value as a convenience.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTag:` (page 597)

Related Sample Code
EnhancedDataBurn

MyCustomColorPicker
Quartz Composer WWDC 2005 TextEdit
VertexPerformanceTest

Declared In
NSCell.h

takeDoubleValueFrom:

Sets the value of the receiver's cell to a double-precision floating-point value obtained from the specified object.

```
- (void)takeDoubleValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the `doubleValue` (page 552) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setDoubleValue:` (page 584)

Declared In
NSCell.h

takeFloatValueFrom:

Sets the value of the receiver's cell to a single-precision floating-point value obtained from the specified object.

```
- (void)takeFloatValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the `floatValue` (page 557) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setFloatValue:` (page 587)

Declared In
NSCell.h

takeIntegerValueFrom:

Sets the value of the receiver's cell to an integer value obtained from the specified object.

```
- (void)takeIntegerValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the `integerValue` (page 564) message.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setIntValue:` (page 590)
- `setIntegerValue:` (page 590)

Declared In
`NSCell.h`

takeIntValueFrom:

Sets the value of the receiver's cell to an integer value obtained from the specified object.

```
- (void)takeIntValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the `intValue` (page 565) message.

Discussion
On Mac OS X v10.5 and later you should use `takeIntegerValueFrom:` (page 606) instead.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `takeIntegerValueFrom:` (page 606)
- `setIntValue:` (page 590)
- `setIntegerValue:` (page 590)

Declared In
`NSCell.h`

takeObjectValueFrom:

Sets the value of the receiver's cell to the object value obtained from the specified object.

```
- (void)takeObjectValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the `objectValue` (page 572) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setObjectValue:` (page 592)

Declared In
`NSCell.h`

takeStringValueFrom:

Sets the value of the receiver's cell to the string value obtained from the specified object.

```
- (void)takeStringValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the `stringValue` (page 605) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setStringValue:` (page 596)

Declared In
`NSCell.h`

target

Returns the target object of the receiver.

```
- (id)target
```

Return Value
The target object that receives action messages from the cell. The `NSCell` implementation of this method returns `nil`.

Discussion
Subclasses (such as `NSActionCell`) override this method to return the target object as part of the target/action implementation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTarget:` (page 598)

Declared In
`NSCell.h`

title

Returns the receiver's title.

```
- (NSString *)title
```

Return Value
The cell's string value.

Discussion
Subclasses (such as `NSButtonCell`) may override this method to return a different value.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitle:` (page 598)

Related Sample Code
PhotoSearch
SourceView

Declared In
`NSCell.h`

titleRectForBounds:

Returns the rectangle in which the receiver draws its title text.

```
- (NSRect)titleRectForBounds:(NSRect)theRect
```

Parameters
theRect
The bounding rectangle of the receiver.

Return Value
The rectangle in which the receiver draws its title text.

Discussion
If the receiver is a text-type cell, this method resizes the drawing rectangle for the title (*theRect*) inward by a small offset to accommodate the cell border. If the receiver is not a text-type cell, the method does nothing.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- imageRectForBounds:` (page 563)

Related Sample Code
PhotoSearch

Declared In
`NSCell.h`

trackMouse:inRect:ofView:untilMouseUp:

Initiates the mouse tracking behavior in a cell.

```
- (BOOL)trackMouse:(NSEvent *)theEvent inRect:(NSRect)cellFrame ofView:(NSView
    *)controlView untilMouseUp:(BOOL)untilMouseUp
```

Parameters

theEvent
The event that caused the mouse tracking to occur.

cellFrame
The receiver's frame rectangle.

controlView
The view containing the receiver. This is usually an `NSControl` object.

untilMouseUp
If `YES`, mouse tracking continues until the user releases the mouse button. If `NO`, tracking continues until the cursor leaves the tracking rectangle, specified by the *cellFrame* parameter, regardless of the mouse button state. See the discussion for more information.

Return Value
`YES` if the mouse tracking conditions are met, otherwise `NO`.

Discussion
This method is generally not overridden because the default implementation invokes other `NSCell` methods that can be overridden to handle specific events in a dragging session. This method's return value depends on the *untilMouseUp* flag. If *untilMouseUp* is set to `YES`, this method returns `YES` if the mouse button goes up while the cursor is anywhere; `NO`, otherwise. If *untilMouseUp* is set to `NO`, this method returns `YES` if the mouse button goes up while the cursor is within *cellFrame*; `NO`, otherwise.

This method first invokes `startTrackingAt:inView:` (page 603). If that method returns `YES`, then as mouse-dragged events are intercepted, `continueTracking:at:inView:` (page 550) is invoked until either the method returns `NO` or the mouse is released. Finally, `stopTracking:at:inView:mouseIsUp:` (page 604) is invoked if the mouse is released. If *untilMouseUp* is `YES`, it's invoked when the mouse button goes up while the cursor is anywhere. If *untilMouseUp* is `NO`, it's invoked when the mouse button goes up while the cursor is within *cellFrame*. You usually override one or more of these methods to respond to specific mouse events.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSCell.h`

truncatesLastVisibleLine

Returns a Boolean value indicating whether the receiver truncates and adds the ellipsis character to the last visible line if the text doesn't fit into the cell bounds.

```
- (BOOL)truncatesLastVisibleLine
```

Return Value
`YES` if the receiver truncates the last line; otherwise `NO`.

Discussion
The line break mode must be either `NSLineBreakByWordWrapping` or `NSLineBreakByCharWrapping`. Otherwise, this setting is ignored.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setTruncatesLastVisibleLine:` (page 599)
- `lineBreakMode` (page 570)

Declared In
`NSCell.h`

type

Returns the type of the receiver

```
- (NSCellType)type
```

Return Value
The type of the cell (see "`NSCellType`" (page 613) for possible values).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setType:` (page 599)

Related Sample Code
ClockControl
PhotoSearch

Declared In
`NSCell.h`

userInterfaceLayoutDirection

Returns the layout direction of the user interface.

```
- (NSUserInterfaceLayoutDirection)userInterfaceLayoutDirection
```

Return Value
The direction of the user interface layout. See `NSUserInterfaceLayoutDirection` (page 184) for possible values.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setUserInterfaceLayoutDirection:` (page 600)

Declared In
NSCell.h

usesSingleLineMode

Returns whether the text cell restricts layout and rendering of its content to a single line.

```
- (BOOL)usesSingleLineMode
```

Return Value
YES if layout and rendering is restricted to a single line, otherwise NO.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setUsesSingleLineMode: (page 601)

Declared In
NSCell.h

wantsNotificationForMarkedText

Returns a Boolean value that indicates whether the field editor initiated by the receiver should post text change notifications.

```
- (BOOL)wantsNotificationForMarkedText
```

Return Value
YES if the field editor initiated by the receiver should post text change notifications (NSTextDidChangeNotification (page 2752)) while editing marked text; otherwise, they are delayed until the marked text confirmation.

Discussion
NSCell's implementation returns NO.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSCell.h

wraps

Returns a Boolean value that indicates whether the receiver wraps its text when the text exceeds the borders of the cell.

```
- (BOOL)wraps
```

Return Value
YES if the receiver wraps text, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setWraps:` (page 601)

Declared In
`NSCell.h`

Constants

NSCellType

These constants specify how a cell represents its data (as text or as an image). These constants are used by `setType:` (page 599) and `type` (page 611).

```
enum {
    NSNullCellType  = 0,
    NSTextCellType  = 1,
    NSImageCellType = 2
};
typedef NSUInteger NSCellType;
```

Constants

`NSNullCellType`
Cell displays nothing.
Available in Mac OS X v10.0 and later.
Declared in `NSCell.h`.

`NSTextCellType`
Cell displays text.
Available in Mac OS X v10.0 and later.
Declared in `NSCell.h`.

`NSImageCellType`
Cell displays images.
Available in Mac OS X v10.0 and later.
Declared in `NSCell.h`.

NSCellAttribute

These constants specify how a button behaves when pressed and how it displays its state. These constants are used by the `NSButton` and `NSButtonCell` classes

```
enum {
    NSCellDisabled               = 0,
    NSCellState                  = 1,
    NSPushInCell                 = 2,
    NSCellEditable               = 3,
    NSChangeGrayCell             = 4,
    NSCellHighlighted            = 5,
    NSCellLightsByContents       = 6,
    NSCellLightsByGray           = 7,
    NSChangeBackgroundCell       = 8,
    NSCellLightsByBackground     = 9,
    NSCellIsBordered             = 10,
    NSCellHasOverlappingImage    = 11,
    NSCellHasImageHorizontal     = 12,
    NSCellHasImageOnLeftOrBottom = 13,
    NSCellChangesContents        = 14,
    NSCellIsInsetButton          = 15,
    NSCellAllowsMixedState       = 16
};
typedef NSUInteger NSCellAttribute;
```

Constants

`NSCellAllowsMixedState`

Lets the cell's state be `NSMixedState`, as well as `NSOffState` and `NSOnState`.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSChangeBackgroundCell`

If the cell's state is `NSMixedState` or `NSOnState`, changes the cell's background color from gray to white.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellChangesContents`

If the cell's state is `NSMixedState` or `NSOnState`, displays the cell's alternate image.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSChangeGrayCell`

If the cell's state is `NSMixedState` or `NSOnState`, displays the cell's image as darkened.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellDisabled`

Does not let the user manipulate the cell.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellEditable`

Lets the user edit the cell's contents.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellHasImageHorizontal`
Controls the position of the cell's image: places the image on the right of any text in the cell.

Together, `NSCellHasImageOnLeftOrBottom`, `NSCellHasImageHorizontal`, and `NSCellHasOverlappingImage` control the position of the cell's image and text. To place the image above, set none of them. To place the image below, set `NSCellHasImageOnLeftOrBottom`. To place the image to the right, set `NSCellHasImageHorizontal`. To place the image to the left, set `NSCellHasImageHorizontal` and `NSCellHasImageOnLeftOrBottom`. To place the image directly over, set `NSCellHasOverlappingImage`.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellHasImageOnLeftOrBottom`
Controls the position of the cell's image: places the image on the left of or below any text in the cell.

See `NSCellHasImageHorizontal` (page 615) for more details.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellHasOverlappingImage`
Controls the position of the cell's image: places the image over any text in the cell.

See `NSCellHasImageHorizontal` (page 615) for more details.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellHighlighted`

Draws the cell with a highlighted appearance. (Deprecated. Use `setHighlighted:` (page 588) instead.)

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellIsBordered`
Draws a border around the cell.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellIsInsetButton`
Insets the cell's contents from the border.

By default, the cell's contents are inset by 2 points. This constant is ignored if the cell is unbordered.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellLightsByBackground`
If the cell is pushed in, changes the cell's background color from gray to white.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellLightsByContents`
If the cell is pushed in, displays the cell's alternate image.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellLightsByGray`
If the cell is pushed in, displays the cell's image as darkened.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSPushInCell`
Determines whether the cell's image and text appear to be shifted down and to the right.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSCellState`
The cell's state.

The cell's state can be `NSMixedState` (page 618), `NSOffState` (page 618), or `NSOnState` (page 618).

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

NSCellImagePosition

These constants specify the position of a button's image relative to its title. These constants are used by the `setImagePosition:` (page 483) and `imagePosition` (page 476) methods of `NSButton` and `NSButtonCell`.

```
enum {
    NSNoImage       = 0,
    NSImageOnly     = 1,
    NSImageLeft     = 2,
    NSImageRight    = 3,
    NSImageBelow    = 4,
    NSImageAbove    = 5,
    NSImageOverlaps = 6
};
typedef NSUInteger NSCellImagePosition;
```

Constants

`NSNoImage`
The cell doesn't display an image.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSImageOnly`
The cell displays an image, but not a title.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSImageLeft`
The image is to the left of the title.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSImageRight`
The image is to the right of the title.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSImageBelow`
The image is below the title.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSImageAbove`
The image is above the title.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSImageOverlaps`
The image overlaps the title.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

NSImageScaling

These constants specify a cell's image scaling behavior.

```
enum {
    NSImageScaleProportionallyDown = 0,
    NSImageScaleAxesIndependently,
    NSImageScaleNone,
    NSImageScaleProportionallyUpOrDown
};
typedef NSUInteger NSImageScaling;
```

Constants

`NSImageScaleProportionallyDown`
If it is too large for the destination, scale the image down while preserving the aspect ratio.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

`NSImageScaleAxesIndependently`
Scale each dimension to exactly fit destination.

This setting does not preserve the aspect ratio of the image.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

`NSImageScaleNone`
Do not scale the image.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

`NSImageScaleProportionallyUpOrDown`
Scale the image to its maximum possible dimensions while both staying within the destination area and preserving its aspect ratio.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

NSCellStateValue

These constants specify a cell's state and are used mostly for buttons. These constants are described in Cell States of *Control and Cell Programming Topics for Cocoa*.

```
enum {
    NSMixedState = -1,
    NSOffState   = 0,
    NSOnState    = 1
};
typedef NSUInteger NSCellStateValue;
```

Constants

`NSMixedState`

The corresponding feature is in effect somewhere.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSOffState`

The corresponding feature is in effect nowhere.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSOnState`

The corresponding feature is in effect everywhere.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

State Masks

These constants specify what happens when a button is pressed or is displaying its alternate state. These contents are used by the `highlightsBy` (page 501) and `showsStateBy` (page 520) methods of `NSButtonCell`.

```
enum {
   NSNoCellMask               = 0,
   NSContentsCellMask         = 1,
   NSPushInCellMask           = 2,
   NSChangeGrayCellMask       = 4,
   NSChangeBackgroundCellMask = 8
};
```

Constants

`NSNoCellMask`

The button cell doesn't change.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSPushInCellMask`

The button cell "pushes in" if it has a border.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSContentsCellMask`
The button cell displays its alternate icon and/or title.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSChangeGrayCellMask`
The button cell swaps the "control color" (the `controlColor` (page 678) method of `NSColor`) and white pixels on its background and icon.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSChangeBackgroundCellMask`
Same as `NSChangeGrayCellMask`, but only background pixels are changed.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

NSControlTint

These constants specify a cell's tint. These constants are used by `controlTint` (page 551) and `setControlTint:` (page 583).

```
enum {
    NSDefaultControlTint  = 0,
    NSBlueControlTint     = 1,
    NSGraphiteControlTint = 6,
    NSClearControlTint    = 7
};
typedef NSUInteger NSControlTint;
```

Constants

`NSDefaultControlTint`
The current default tint setting

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSClearControlTint`
Clear control tint

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSBlueControlTint`
Aqua control tint

Available in Mac OS X v10.3 and later.

Declared in `NSCell.h`.

`NSGraphiteControlTint`
Graphite control tint

Available in Mac OS X v10.3 and later.

Declared in `NSCell.h`.

NSControlSize

These constants specify a cell's size. These constants are used by `controlSize` (page 551) and `setControlSize:` (page 582).

```
enum {
    NSRegularControlSize,
    NSSmallControlSize,
    NSMiniControlSize
};
typedef NSUInteger NSControlSize;
```

Constants

`NSRegularControlSize`
The control is sized as regular.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSSmallControlSize`
The control has a smaller size.

This constant is for controls that cannot be resized in one direction, such as push buttons, radio buttons, checkboxes, sliders, scroll bars, pop-up buttons, tabs, and progress indicators. You should use a small system font with a small control.

Available in Mac OS X v10.0 and later.

Declared in `NSCell.h`.

`NSMiniControlSize`
The control has a smaller size than `NSSmallControlSize`.

Available in Mac OS X v10.3 and later.

Declared in `NSCell.h`.

Hit Testing

These constants are used by `hitTestForEvent:inRect:ofView:` (page 561) to determine the effect of an event.

```
enum {
    NSCellHitNone = 0,
    NSCellHitContentArea = 1 << 0,
    NSCellHitEditableTextArea = 1 << 1,
    NSCellHitTrackableArea = 1 << 2,
};
```

Constants

`NSCellHitNone`
An empty area, or did not hit in the cell.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

`NSCellHitContentArea`
A content area in the cell.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

`NSCellHitEditableTextArea`
An editable text area of the cell.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

`NSCellHitTrackableArea`
A trackable area in the cell.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

Declared In
`NSCell.h`

NSBackgroundStyle

Background styles used with `backgroundStyle` (page 547), `setBackgroundStyle:` (page 580), and `interiorBackgroundStyle` (page 565).

```
enum {
    NSBackgroundStyleLight = 0,
    NSBackgroundStyleDark,
    NSBackgroundStyleRaised,
    NSBackgroundStyleLowered
};
typedef NSUInteger NSBackgroundStyle;
```

Constants
`NSBackgroundStyleLight`
The background is a light color.

Dark content contrasts well with this background.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

`NSBackgroundStyleDark`
The background is a dark color.

Light content contrasts well with this background.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

`NSBackgroundStyleRaised`
The background is intended to appear higher than the content drawn on it.

Content might need to be inset.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

`NSBackgroundStyleLowered`
The background is intended to appear lower than the content drawn on it.

Content might need to be embossed.

Available in Mac OS X v10.5 and later.

Declared in `NSCell.h`.

Deprecated Scaling Constants

These are deprecated scaling constants. (Deprecated. Use "NSImageScaling" (page 617) constants instead.)

```
enum {
    NSScaleProportionally = 0,
    NSScaleToFit,
    NSScaleNone
};
```

Constants

`NSScaleProportionally`
Use `NSImageScaleProportionallyDown` (page 617).

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

Declared in `NSCell.h`.

`NSScaleToFit`
Use `NSImageScaleAxesIndependently` (page 617).

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

Declared in `NSCell.h`.

`NSScaleNone`
Use `NSImageScaleNone` (page 617).

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

Declared in `NSCell.h`.

Data Entry Types

These constants specify how a cell formats numeric data.

```
enum {
   NSAnyType            = 0,
   NSIntType            = 1,
   NSPositiveIntType    = 2,
   NSFloatType          = 3,
   NSPositiveFloatType  = 4,
   NSDoubleType         = 6,
   NSPositiveDoubleType = 7
};
```

Constants

`NSIntType`
Must be between `INT_MIN` and `INT_MAX`.
Deprecated in Mac OS X v10.4 and later.
Declared in `NSCell.h`.

`NSPositiveIntType`
Must be between 1 and `INT_MAX`.
Deprecated in Mac OS X v10.4 and later.
Declared in `NSCell.h`.

`NSFloatType`
Must be between `-FLT_MAX` and `FLT_MAX`.
Deprecated in Mac OS X v10.4 and later.
Declared in `NSCell.h`.

`NSPositiveFloatType`
Must be between `FLT_MIN` and `FLT_MAX`.
Deprecated in Mac OS X v10.4 and later.
Declared in `NSCell.h`.

`NSDoubleType`
Must be between `-FLT_MAX` and `FLT_MAX`.
Deprecated in Mac OS X v10.4 and later.
Declared in `NSCell.h`.

`NSPositiveDoubleType`
Must be between `FLT_MIN` and `FLT_MAX`.
Deprecated in Mac OS X v10.4 and later.
Declared in `NSCell.h`.

`NSAnyType`
Any value is allowed.
Deprecated in Mac OS X v10.4 and later.
Declared in `NSCell.h`.

Discussion

These constants are used by `setEntryType:` (page 585) and `entryType` (page 556).

Declared In

`NSCell.h`

Notifications

NSControlTintDidChangeNotification

Sent after the user changes control tint preference. The notification object is `NSApp`. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSCell.h`

NSCIImageRep Class Reference

Inherits from	NSImageRep : NSObject
Conforms to	NSCoding (NSImageRep) NSCopying (NSImageRep) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	AppKit/NSCIImageRep.h
Companion guide	Cocoa Drawing Guide
Related sample code	FunHouse Reducer StillMotion

Overview

An `NSCIImageRep` object can render an image from a Core Image `CIImage` instance. For more information about the `CIImage` class and Core Image, see *Core Image Programming Guide*.

Tasks

Initialization

`+ imageRepWithCIImage:` (page 626)
Creates and returns an `NSCIImageRep` object initialized to the specified `CIImage` instance.

`- initWithCIImage:` (page 626)
Returns an `NSCIImageRep` object initialized to the specified `CIImage` instance.

Returning an Image

`- CIImage` (page 626)
Returns the receiver's `CIImage` instance.

Class Methods

imageRepWithCIImage:

Creates and returns an `NSCIImageRep` object initialized to the specified `CIImage` instance.

```
+ (id)imageRepWithCIImage:(CIImage *)image
```

Parameters

image

The `CIImage` instance.

Return Value

An initialized `NSCIImageRep` object, or `nil` if the object could not be initialized.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

FunHouse

Reducer

StillMotion

Declared In

`NSCIImageRep.h`

Instance Methods

CIImage

Returns the receiver's `CIImage` instance.

```
- (CIImage *)CIImage
```

Return Value

The `CIImage` instance.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSCIImageRep.h`

initWithCIImage:

Returns an `NSCIImageRep` object initialized to the specified `CIImage` instance.

```
- (id)initWithCIImage:(CIImage *)image
```

Parameters

image

The `CIImage` instance.

Return Value

An initialized `NSCIImageRep` object, or `nil` if the object could not be initialized.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSCIImageRep.h`

NSClipView Class Objective-C Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSClipView.h
Companion guide	Cocoa Drawing Guide
Related sample code	Quartz Composer WWDC 2005 TextEdit TextSizingExample

Class at a Glance

An NSClipView contains and scrolls the document view displayed by an NSScrollView. You normally don't need to program with NSClipViews, as NSScrollView handles most of the details of their operation.

Principal Attributes

- Efficient scrolling by copying drawn portions of the document view
- Monitoring of document view for automatic update

Interface Builder

`- initWithFrame:`
: Initializes the NSClipView.

Commonly Used Methods

`setDocumentView:` (page 636)
: Sets the view scrolled within the NSClipView.

`setCopiesOnScroll:` (page 636)
: Sets whether the NSClipView copies drawn portions of the document view during scrolling.

Overview

An NSClipView holds the document view of an NSScrollView, clipping the document view to its frame, handling the details of scrolling in an efficient manner, and updating the NSScrollView when the document view's size or position changes. You don't normally use the NSClipView class directly; it's provided primarily as the scrolling machinery for the NSScrollView class. However, you might use the NSClipView class to implement a class similar to NSScrollView.

Interaction With NSScrollView

When using an NSClipView within an NSScrollView (the usual configuration), you should issue messages that control background drawing state to the NSScrollView, rather than messaging the NSClipView directly. This recommendation applies to the following methods:

- `-setBackgroundColor:`
- `-backgroundColor`
- `-setDrawsBackground:`
- `-drawsBackground`

The NSClipView methods are intended for when the NSClipView is used independently of a containing NSScrollView. In the usual case, NSScrollView should be allowed to manage the background-drawing properties of its associated NSClipView.

There is only one background-drawing state per NSScrollView/NSClipView pair. The two objects do not maintain independent and distinct `drawsBackground` and `backgroundColor` properties; rather, NSScrollView's accessors for these properties largely defer to the associated NSClipView and allow the NSClipView to maintain the state. In Mac OS X v10.2 and earlier system versions, NSScrollView maintained a cache of the last state it set for its NSClipView. If the NSClipView was sent a `setDrawsBackground:` message directly, the cache might not reflect the state accurately. This caching of state has been removed in Mac OS X v10.3.

It is also important to note that sending a `setDrawsBackground:` message with a parameter of NO to an NSScrollView has the added effect of sending the NSClipView a `setCopiesOnScroll:` message with a parameter of NO. The side effect of sending the `setDrawsBackground:` message directly to the NSClipView is the appearance of "trails" (vestiges of previous drawing) in the document view as it is scrolled.

Tasks

Setting the Document View

`- setDocumentView:` (page 636)
Sets the receiver's document view to *`aView`*, removing any previous document view, and sets the origin of the receiver's bounds rectangle to the origin of *`aView`*'s frame rectangle.

`- documentView` (page 634)
Returns the receiver's document view.

Scrolling

- `scrollToPoint:` (page 635)
  Changes the origin of the receiver's bounds rectangle to *`newOrigin`*.
- `autoscroll:` (page 632)
  Scrolls the receiver proportionally to *`theEvent`*'s distance outside of it.
- `constrainScrollPoint:` (page 633)
  Returns a scroll point adjusted from *`proposedNewOrigin`*, if necessary, to guarantee the receiver will still lie within its document view.

Determining Scrolling Efficiency

- `setCopiesOnScroll:` (page 636)
  Controls whether the receiver copies rendered images while scrolling.
- `copiesOnScroll` (page 633)
  Returns `YES` if the receiver copies its existing rendered image while scrolling (only drawing exposed portions of its document view), `NO` if it forces its contents to be redrawn each time.

Getting the Visible Portion

- `documentRect` (page 634)
  Returns the rectangle defining the document view's frame, adjusted to the size of the receiver if the document view is smaller.
- `documentVisibleRect` (page 634)
  Returns the exposed rectangle of the receiver's document view, in the document view's own coordinate system.

Setting the Document Cursor

- `setDocumentCursor:` (page 636)
  Sets the cursor object used over the receiver to *`aCursor`*.
- `documentCursor` (page 633)
  Returns the cursor object used when the cursor lies over the receiver.

Working with Background Color

- `drawsBackground` (page 635)
  Returns `YES` if the receiver draws its background color.
- `setDrawsBackground:` (page 637)
  Sets whether the receiver draws its background color, depending on the Boolean value *`flag`*.
- `setBackgroundColor:` (page 635)
  Sets the receiver's background color to *`aColor`*.

- `backgroundColor` (page 632)
  Returns the color of the receiver's background.

Overriding NSView Methods

- `viewBoundsChanged:` (page 637)
  Handles an `NSViewBoundsDidChangeNotification` (page 3256), passed in the *`aNotification`* argument, by updating a containing NSScrollView based on the new bounds.
- `viewFrameChanged:` (page 638)
  Handles an `NSViewFrameDidChangeNotification` (page 3257), passed in the *`aNotification`* argument, by updating a containing NSScrollView based on the new frame.

Instance Methods

autoscroll:

Scrolls the receiver proportionally to *`theEvent`*'s distance outside of it.

```
- (BOOL)autoscroll:(NSEvent *)theEvent
```

Discussion
`theEvent`'s location should be expressed in the window's base coordinate system (which it normally is), not the receiving NSClipView's. Returns `YES` if any scrolling is performed; otherwise returns `NO`.

Never invoke this method directly; instead, the NSClipView's document view should repeatedly send itself `autoscroll:` (page 3148) messages when the cursor is dragged outside the NSClipView's frame during a modal event loop initiated by a mouse-down event. The NSView class implements `autoscroll:` (page 3148) to forward the message to the receiver's superview; thus the message is ultimately forwarded to the NSClipView.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSClipView.h`

backgroundColor

Returns the color of the receiver's background.

```
- (NSColor *)backgroundColor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBackgroundColor:` (page 635)

Declared In
NSClipView.h

constrainScrollPoint:

Returns a scroll point adjusted from *proposedNewOrigin*, if necessary, to guarantee the receiver will still lie within its document view.

```
- (NSPoint)constrainScrollPoint:(NSPoint)proposedNewOrigin
```

Discussion
For example, if *proposedNewOrigin*'s y coordinate lies to the left of the document view's origin, then the y coordinate returned is set to that of the document view's origin.

Availability
Available in Mac OS X v10.0 and later.

See Also
- scrollToPoint: (page 635)

Declared In
NSClipView.h

copiesOnScroll

Returns YES if the receiver copies its existing rendered image while scrolling (only drawing exposed portions of its document view), NO if it forces its contents to be redrawn each time.

```
- (BOOL)copiesOnScroll
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- setCopiesOnScroll: (page 636)

Declared In
NSClipView.h

documentCursor

Returns the cursor object used when the cursor lies over the receiver.

```
- (NSCursor *)documentCursor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- setDocumentCursor: (page 636)

Declared In
NSClipView.h

documentRect

Returns the rectangle defining the document view's frame, adjusted to the size of the receiver if the document view is smaller.

```
- (NSRect)documentRect
```

Discussion
In other words, this rectangle is always at least as large as the receiver itself.

The document rectangle is used in conjunction with an NSClipView's bounds rectangle to determine values for the indicators of relative position and size between the NSClipView and its document view. For example, NSScrollView uses these rectangles to set the size and position of the knobs in its scrollers. When the document view is much larger than the NSClipView, the knob is small; when the document view is near the same size, the knob is large; and when the document view is the same size or smaller, there is no knob.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `reflectScrolledClipView:` (page 2351) (NSScrollView)
- `documentVisibleRect` (page 634)

Declared In
NSClipView.h

documentView

Returns the receiver's document view.

```
- (id)documentView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDocumentView:` (page 636)

Declared In
NSClipView.h

documentVisibleRect

Returns the exposed rectangle of the receiver's document view, in the document view's own coordinate system.

```
- (NSRect)documentVisibleRect
```

Discussion
Note that this rectangle doesn't reflect the effects of any clipping that may occur above the NSClipView itself. To get the portion of the document view that's guaranteed to be visible, send it a `visibleRect` message.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- documentRect` (page 634)

Declared In
`NSClipView.h`

drawsBackground

Returns `YES` if the receiver draws its background color.

```
- (BOOL)drawsBackground
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setDrawsBackground:` (page 637)

Declared In
`NSClipView.h`

scrollToPoint:

Changes the origin of the receiver's bounds rectangle to *newOrigin*.

```
- (void)scrollToPoint:(NSPoint)newOrigin
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- constrainScrollPoint:` (page 633)

Related Sample Code
WhackedTV

Declared In
`NSClipView.h`

setBackgroundColor:

Sets the receiver's background color to *aColor*.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `backgroundColor` (page 632)

Declared In
`NSClipView.h`

setCopiesOnScroll:

Controls whether the receiver copies rendered images while scrolling.

```
- (void)setCopiesOnScroll:(BOOL)flag
```

Discussion
If *flag* is `YES`, the receiver copies the existing rendered image to its new location while scrolling and only draws exposed portions of its document view. If *flag* is `NO`, the receiver always forces its document view to draw itself on scrolling.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `copiesOnScroll` (page 633)

Declared In
`NSClipView.h`

setDocumentCursor:

Sets the cursor object used over the receiver to *aCursor*.

```
- (void)setDocumentCursor:(NSCursor *)aCursor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `documentCursor` (page 633)

Declared In
`NSClipView.h`

setDocumentView:

Sets the receiver's document view to *aView*, removing any previous document view, and sets the origin of the receiver's bounds rectangle to the origin of *aView*'s frame rectangle.

```
- (void)setDocumentView:(NSView *)aView
```

Discussion
If the receiver is contained in an NSScrollView, you should send the NSScrollView a `setDocumentView:` (page 2355) message instead, so it can perform whatever updating it needs.

In the process of setting the document view, this method registers the receiver for the notifications `NSViewFrameDidChangeNotification` (page 3257) and `NSViewBoundsDidChangeNotification` (page 3256), adjusts the key view loop to include the new document view, and updates a parent NSScrollView's display if needed using `reflectScrolledClipView:` (page 2351).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `documentView` (page 634)

Related Sample Code
TextSizingExample

Declared In
`NSClipView.h`

setDrawsBackground:

Sets whether the receiver draws its background color, depending on the Boolean value *flag*.

```
- (void)setDrawsBackground:(BOOL)flag
```

Discussion
If your NSClipView is enclosed in an NSScrollView, you should send the `setDrawsBackground:` message to the NSScrollView. Sending a `setDrawsBackground:` message with a parameter of NO to an NSScrollView has the added effect of sending the NSClipView a `setCopiesOnScroll:` message with a parameter of NO. The side effect of sending the `setDrawsBackground:` message directly to the NSClipView is the appearance of "trails" (vestiges of previous drawing) in the document view as it is scrolled.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawsBackground` (page 635)

Declared In
`NSClipView.h`

viewBoundsChanged:

Handles an `NSViewBoundsDidChangeNotification` (page 3256), passed in the *aNotification* argument, by updating a containing NSScrollView based on the new bounds.

```
- (void)viewBoundsChanged:(NSNotification *)aNotification
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSClipView.h

viewFrameChanged:

Handles an NSViewFrameDidChangeNotification (page 3257), passed in the *aNotification* argument, by updating a containing NSScrollView based on the new frame.

```
- (void)viewFrameChanged:(NSNotification *)aNotification
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSClipView.h

NSCoder Application Kit Additions Reference

Inherits from	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSColor.h
Companion guide	Archives and Serializations Programming Guide

Overview

This category adds a single method to the Foundation framework's NSCoder class. This method, `decodeNXColor` (page 639), is used to convert archived `NXColor` structures into `NSColor` objects.

`NXColor`, a type that dates from pre-OpenStep versions of NEXTSTEP, was a `struct`. Its replacement, `NSColor`, is a class. The difficulties of converting from a `struct` to a class require a special method like `decodeNXColor` (page 639).

The `decodeNXColor` (page 639) method becomes part of the `NSCoder` class only for applications that use the Application Kit.

Tasks

Decoding NXColor Structures

- `decodeNXColor` (page 639)
  Decodes a color structure from NEXTSTEP Release 3 or earlier and returns an `NSColor` object.

Instance Methods

decodeNXColor

Decodes a color structure from NEXTSTEP Release 3 or earlier and returns an `NSColor` object.

```
- (NSColor *)decodeNXColor
```

Return Value
An autoreleased `NSColor` object. Returns `nil` if the archived color is invalid.

Discussion
This method does not have a matching method for encoding an `NXColor` structure. Encode an `NSColor` object instead.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColor.h`

NSCollectionView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSCollectionView.h
Companion guide	Collection View Programming Guide
Related sample code	IconCollection Reviews

Overview

`NSCollectionView` class displays an array of content as a grid of views. The views are specified using the `NSCollectionViewItem` class which makes loadings nibs containing the view easy, and supports bindings.

Tasks

Modifying the Collection View Item

`- setItemPrototype:` (page 651)
Sets the receiver's item prototype to the specified collection view item.

`- itemPrototype` (page 647)
Returns the receiver's collection view item prototype.

`- newItemForRepresentedObject:` (page 648)
Returns the collection view item that is used for the specified object.

Working with the Responder Chain

- `isFirstResponder` (page 646)
  Returns whether the receiver is the first responder.

Setting the Content

- `setContent:` (page 650)
  Sets the receiver's content array.
- `content` (page 644)
  Returns the receiver's content object.

Setting the Selection Mode

- `setSelectable:` (page 653)
  Controls whether the receiver allows the user to select items.
- `isSelectable` (page 646)
  Returns a Boolean value that indicates whether the receiver allows the user to select items, `NO` if it doesn't.
- `setAllowsMultipleSelection:` (page 649)
  Controls whether the user can select multiple items at a time.
- `allowsMultipleSelection` (page 643)
  Returns a Boolean value that indicates whether the receiver allows the user to select more than one item at a time.
- `setSelectionIndexes:` (page 653)
  Sets the receiver's selection using the specified indexes.
- `selectionIndexes` (page 649)
  Returns an index set containing the indexes of the receiver's currently selected objects in the content array.

Laying out the Collection View

- `setMaxNumberOfRows:` (page 652)
  Sets the maximum number of rows the receiver will display.
- `maxNumberOfRows` (page 648)
  Returns the maximum number of rows the receiver will display.
- `setMaxNumberOfColumns:` (page 652)
  Sets the maximum number of columns the receiver will display
- `maxNumberOfColumns` (page 647)
  Returns the maximum number of columns the receiver will display.
- `setMinItemSize:` (page 653)
  Sets the minimum size used to display individual layout items in the grid.
- `minItemSize` (page 648)
  Returns the minimum size used to display individual collection view items in the grid.

- `setMaxItemSize:` (page 651)
  Sets the maximum size used to display individual collection view items in the grid.
- `maxItemSize` (page 647)
  Returns the maximum size used to display individual collection view items in the grid

Modifying the Background

- `setBackgroundColors:` (page 649)
  Sets the receiver's background colors to the specified array of colors.
- `backgroundColors` (page 644)
  Return the receiver's background colors.

Getting and Setting the Delegate

- `delegate` (page 644)
  Returns the receiver's delegate.
- `setDelegate:` (page 650)
  Sets the receiver's delegate.

Drag and Drop Support

- `draggingImageForItemsAtIndexes:withEvent:offset:` (page 645)
  This method computes and returns an image to use for dragging.
- `setDraggingSourceOperationMask:forLocal:` (page 651)
  Configures the default value returned from `draggingSourceOperationMaskForLocal:` (page 3670).

Getting a Collection Item and Its Frame

- `itemAtIndex:` (page 646)
  Returns the collection view item for the represented object at the specified index.
- `frameForItemAtIndex:` (page 645)
  Returns the frame of the collection view item at the specified index.

Instance Methods

allowsMultipleSelection

Returns a Boolean value that indicates whether the receiver allows the user to select more than one item at a time.

```
- (BOOL)allowsMultipleSelection
```

Return Value
`YES` if the receiver allows the user to select more than one column or row at a time, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

backgroundColors

Return the receiver's background colors.

```
- (NSArray *)backgroundColors
```

Return Value
Returns an array containing the receiver's background colors.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

content

Returns the receiver's content object.

```
- (NSArray *)content
```

Return Value
An array containing the receiver's content.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
IconCollection

Declared In
`NSCollectionView.h`

delegate

Returns the receiver's delegate.

```
- (id < NSCollectionViewDelegate >)delegate
```

Return Value
The receiver's delegate object.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSCollectionView.h

draggingImageForItemsAtIndexes:withEvent:offset:

This method computes and returns an image to use for dragging.

```
- (NSImage *)draggingImageForItemsAtIndexes:(NSIndexSet *)indexes withEvent:(NSEvent
    *)event offset:(NSPointPointer)dragImageOffset
```

Parameters

indexes
The index set of the items to be dragged.

event
Mouse drag event.

dragImageOffset
An in/out parameter that will initially be set to NSZeroPoint. it can be modified to reposition the returned image. A *dragImageOffset* of NSZeroPoint will cause the image to be centered under the mouse.

Return Value
An image containing a rendering of the visible portions of the views for each item.

Discussion
You can override the default image by subclassing NSCollectionView and overriding this method, or by implementing the collectionView:draggingImageForItemsAtIndexes:withEvent:offset: (page 3609) delegate method, it will be preferred over this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSCollectionView.h

frameForItemAtIndex:

Returns the frame of the collection view item at the specified index.

```
- (NSRect)frameForItemAtIndex:(NSUInteger)index
```

Parameters

index
The index of the collection view item.

Return Value
Returns the frame calculated by the receiver where it intends to place the subview for the NSCollectionViewItem at the given index. The rectangle is returned in the receiver's coordinate system.

Discussion
You can use this method in the `collectionView:draggingImageForItemsAtIndexes:withEvent:offset:` (page 3609) method to determine which views are in the visible portion of the enclosing scroll view.

Overriding this method will have no effect on the receiver's subview layout.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSCollectionView.h`

isFirstResponder

Returns whether the receiver is the first responder.

```
- (BOOL)isFirstResponder
```

Return Value
`YES` if the receiver is the first responder, otherwise `NO`.

Special Considerations
This method is fully key-value observing compliant.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

isSelectable

Returns a Boolean value that indicates whether the receiver allows the user to select items, `NO` if it doesn't.

```
- (BOOL)isSelectable
```

Return Value
`YES` if the receiver allows the user to select items, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

itemAtIndex:

Returns the collection view item for the represented object at the specified index.

```
- (NSCollectionViewItem *)itemAtIndex:(NSUInteger)index
```

Parameters

index

The index of the collection view item.

Return Value
An instance of `NSCollectionViewItem`.

Discussion
Rather than using the `NSCollectionViewItem` instance returned by this method to determine the frame of the collection item's view you should use `frameForItemAtIndex:` (page 645), it is significantly more efficient.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSCollectionView.h`

itemPrototype

Returns the receiver's collection view item prototype.

```
- (NSCollectionViewItem *)itemPrototype
```

Return Value
The receiver's collection view item prototype.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

maxItemSize

Returns the maximum size used to display individual collection view items in the grid

```
- (NSSize)maxItemSize
```

Return Value
The maximum size, measured in points, used to display individual collection view items.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

maxNumberOfColumns

Returns the maximum number of columns the receiver will display.

```
- (NSUInteger)maxNumberOfColumns
```

Return Value
The maximum number of columns the receiver will display.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSCollectionView.h

maxNumberOfRows

Returns the maximum number of rows the receiver will display.

```
- (NSUInteger)maxNumberOfRows
```

Return Value
The maximum number of rows the receiver will display.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSCollectionView.h

minItemSize

Returns the minimum size used to display individual collection view items in the grid.

```
- (NSSize)minItemSize
```

Return Value
The minimum size, measured in points, used to display individual collection view items.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSCollectionView.h

newItemForRepresentedObject:

Returns the collection view item that is used for the specified object.

```
- (NSCollectionViewItem *)newItemForRepresentedObject:(id)object
```

Parameters
object
The content object that the collection view item will represent.

Return Value
An initialized collection view item with the specified object and the appropriate view set. The collection view item should not be autoreleased.

Discussion
Subclasses can override this method if the collection view items are not generated from a prototype or if the prototype view needs to be modified. The subclass is responsible for setting the `view` and `representedObject` (page 657) of the new collection view item.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

selectionIndexes

Returns an index set containing the indexes of the receiver's currently selected objects in the content array.

```
- (NSIndexSet *)selectionIndexes
```

Return Value
An index set containing the indexes of the receiver's currently selected objects in the content array.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

setAllowsMultipleSelection:

Controls whether the user can select multiple items at a time.

```
- (void)setAllowsMultipleSelection:(BOOL)flag
```

Parameters
flag
`YES` to allow the user to select multiple items, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

setBackgroundColors:

Sets the receiver's background colors to the specified array of colors.

```
- (void)setBackgroundColors:(NSArray *)colors
```

Parameters

colors

An array containing the background colors for the receiver.

Discussion

Passing an empty array or `nil` resets the background colors to their default values provided by `controlAlternatingRowBackgroundColors` (page 677).

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSCollectionView.h`

setContent:

Sets the receiver's content array.

```
- (void)setContent:(NSArray *)content
```

Parameters

content

An array containing the receiver's content.

Discussion

The content array can also be provided by creating a binding between the receiver's `NSContentBinding` (page 3709) and an array controller's `arrangedObjects` (page 205) method.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSCollectionView.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSCollectionViewDelegate >)aDelegate
```

Parameters

aDelegate

The delegate object for the receiver. The delegate must conform to the `NSCollectionViewDelegate Protocol` protocol.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSCollectionView.h`

setDraggingSourceOperationMask:forLocal:

Configures the default value returned from `draggingSourceOperationMaskForLocal:` (page 3670).

```
- (void)setDraggingSourceOperationMask:(NSDragOperation)dragOperationMask
    forLocal:(BOOL)localDestination
```

Parameters

dragOperationMask

The types of drag operations allowed.

localDestination

If `YES`, mask applies when the drag destination object is in the same application as the receiver; if `NO`, mask applies when the destination object is outside the receiver's application.

Discussion

By default, this method returns `NSDragOperationEvery` (page 3665) when *localDestination* is `YES` and `NSDragOperationNone` (page 3666) when *localDestination* is `NO`. `NSCollectionView` will save the values you set for each *localDestination* value.

You typically will invoke this method, and not override it.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSCollectionView.h`

setItemPrototype:

Sets the receiver's item prototype to the specified collection view item.

```
- (void)setItemPrototype:(NSCollectionViewItem *)prototype
```

Parameters

prototype

The collection view item used as the prototype by the receiver.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSCollectionView.h`

setMaxItemSize:

Sets the maximum size used to display individual collection view items in the grid.

```
- (void)setMaxItemSize:(NSSize)size
```

Parameters

size

The new maximum size, measured in points, with which to display individual collection view items.

Discussion
Setting the size to (0,0) specifies no maximum grid size. The default is (0.0). If the view in the receiver's collection view item prototype is resizable you should set this to the maximum size that the view should be displayed using.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

setMaxNumberOfColumns:

Sets the maximum number of columns the receiver will display

```
- (void)setMaxNumberOfColumns:(NSUInteger)number
```

Parameters

number
The maximum number of columns the receiver will display.

Discussion
Setting to 0 specifies no maximum number of columns. Defaults to 0.

It is possible for a `NSCollectionView` instance to specify both the maximum number of rows and a maximum number of columns. If the number of content objects exceeds the number of displayable items (*n*=`maxNumberOfRows` * `maxNumberOfColumns`) only the first *n* items of the content array are displayed.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

setMaxNumberOfRows:

Sets the maximum number of rows the receiver will display.

```
- (void)setMaxNumberOfRows:(NSUInteger)number
```

Parameters

number
The maximum number of rows the receiver can display.

Discussion
Setting to 0 specifies no maximum number of rows. Defaults to 0.

It is possible for a `NSCollectionView` instance to specify both the maximum number of rows and a maximum number of columns. If the number of content objects exceeds the number of displayable items (*n*=`maxNumberOfRows` * `maxNumberOfColumns`) only the first *n* items of the content array are displayed.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSCollectionView.h

setMinItemSize:

Sets the minimum size used to display individual layout items in the grid.

```
- (void)setMinItemSize:(NSSize)size
```

Parameters

size

The new minimum size, measured in points, with which to display individual layout items.

Discussion
The default is (0.0). If the view in the receiver's collection view item prototype is resizable you should set this to the minimum size that the view should be displayed using.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSCollectionView.h

setSelectable:

Controls whether the receiver allows the user to select items.

```
- (void)setSelectable:(BOOL)flag
```

Parameters

flag

If flag is YES, the receiver allows the user to select items; if flag is NO, it doesn't.

Discussion
You can set selections programmatically regardless of this setting.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSCollectionView.h

setSelectionIndexes:

Sets the receiver's selection using the specified indexes.

```
- (void)setSelectionIndexes:(NSIndexSet *)indexes
```

Parameters

indexes

The set of selection indexes for the receiver.

Discussion
To select all the receiver's objects, indexes should be an index set with indexes [0...count -1]. To deselect all indexes, pass an empty index set.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

Constants

NSCollectionViewDropOperation

These constants specify if acceptance of a drop should be at the item it is dropped on or before the item. These constants are used by the `collectionView:acceptDrop:index:dropOperation:` (page 3608) and `collectionView:validateDrop:proposedIndex:dropOperation:` (page 3610) methods in `NSCollectionViewDelegate Protocol`

```
enum {     NSCollectionViewDropOn = 0,     NSCollectionViewDropBefore = 1, };
typedef NSInteger NSCollectionViewDropOperation;
```

Constants

`NSCollectionViewDropOn`

The drop occurs at the collection view item to which the item was dragged.

Available in Mac OS X v10.6 and later.

Declared in `NSCollectionView.h`.

`NSCollectionViewDropBefore`

The drop occurs above the collection view item to which the item was dragged..

Available in Mac OS X v10.6 and later.

Declared in `NSCollectionView.h`.

NSCollectionViewItem Class Reference

Inherits from	NSViewController : NSResponder : NSObject
Conforms to	NSCopying NSCoding (NSViewController) NSCoding (NSResponder) NSObject (NSObject) NSEditor (Informal Protocol) (NSViewController) NSEditorRegistration (Informal Protocol) (NSViewController)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSCollectionViewItem.h

Overview

`NSCollectionViewItem` class is a controller that manages the relationship between a compound view and the model object that provides its content. Instances of this class are replicated in a `NSCollectionView`.

Important: In Mac OS X v10.5, the superclass of the NSCollectionViewItem class was NSObject. In Mac OS X v10.6 and later, NSCollectionViewItem is now a subclass of NSViewController. This change was made to improve how the view is replicated within the NSCollectionView. NSCollectionViewItem remains binary compatible with the previous implementation and unarchiving is correctly handled.

Tasks

Setting the Represented Object

`- setRepresentedObject:` (page 657)
Sets the receiver's represented object to the specified model object.

`- representedObject` (page 657)
Returns the receiver's represented object.

Modifying the View

- `setView:` (page 658)
  Sets the view the receiver uses to display its represented object.
- `view` (page 658)
  Returns the view the receiver uses to display its represented object.

Managing the Selection

- `setSelected:` (page 657)
  Sets the selection state of the receiver.
- `isSelected` (page 656)
  Returns the selection state of the receiver.

Parent Collection View

- `collectionView` (page 656)
  Returns the receiver's collection view.

Instance Methods

collectionView

Returns the receiver's collection view.

```
- (NSCollectionView *)collectionView
```

Return Value
The receiver's collection view.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

isSelected

Returns the selection state of the receiver.

```
- (BOOL)isSelected
```

Return Value
`YES` if the receiver is selected, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

representedObject

Returns the receiver's represented object. (Available in Mac OS X v10.5 through Mac OS X v10.5.)

```
- (id)representedObject
```

Return Value
The receiver's represented object.

Availability
Available in Mac OS X v10.5 through Mac OS X v10.5.

Declared In
`NSCollectionView.h`

setRepresentedObject:

Sets the receiver's represented object to the specified model object. (Available in Mac OS X v10.5 through Mac OS X v10.5.)

```
- (void)setRepresentedObject:(id)object
```

Parameters
object
The receiver's model object.

Availability
Available in Mac OS X v10.5 through Mac OS X v10.5.

Declared In
`NSCollectionView.h`

setSelected:

Sets the selection state of the receiver.

```
- (void)setSelected:(BOOL)flag
```

Parameters
flag
`YES` if the receiver is selected, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCollectionView.h`

setView:

Sets the view the receiver uses to display its represented object. (Available in Mac OS X v10.5 through Mac OS X v10.5.)

```
- (void)setView:(NSView *)view
```

Parameters

view

The view that is used to display the receiver's represented object.

Availability
Available in Mac OS X v10.5 through Mac OS X v10.5.

Declared In
NSCollectionView.h

view

Returns the view the receiver uses to display its represented object. (Available in Mac OS X v10.5 through Mac OS X v10.5.)

```
- (NSView *)view
```

Return Value
The view the receiver uses to display its represented object.

Availability
Available in Mac OS X v10.5 through Mac OS X v10.5.

Declared In
NSCollectionView.h

NSColor Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSPasteboardWriting NSPasteboardReading NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSColor.h
Companion guide	Color Programming Topics
Related sample code	Cocoa OpenGL From A View to A Movie From A View to A Picture Quartz Composer WWDC 2005 TextEdit Sketch-112

Class at a Glance

An `NSColor` object represents a color, which is defined in a color space, each point of which has a set of components (such as red, green, and blue) that uniquely define a color.

Principal Attributes

- Color space
- Color components

  Various `colorWith...` and `colorUsing...` methods.
  Preset colors: `blackColor` (page 667), `blueColor` (page 668), and so on.

Commonly Used Methods

`colorUsingColorSpaceName:` (page 700)
Creates an `NSColor` instance in the specified color space.

`set` (page 711)
Sets the drawing color.

Overview

An `NSColor` object represents color and sometimes opacity (alpha). By sending a `set` (page 711) message to an `NSColor` instance, you set the color for the current drawing context. Setting the color causes subsequently drawn graphics to have the color represented by the `NSColor` instance.

It is invalid to use an accessor method related to components of a particular color space on an `NSColor` object that is not in that color space. For example, methods such as `redComponent` (page 710) and `getRed:green:blue:alpha:` (page 705) work on color objects in the calibrated and device RGB color spaces. If you send such a message to an `NSColor` object in the CMYK color space, an exception is raised. Further, the methods `getComponents:` (page 703) and `numberOfComponents` (page 709) work in color spaces that have individual components. Thus they return the components of `NSColor` objects as individual floating-point values regardless of whether they're based on `NSColorSpace` objects or named color spaces. However, older component-fetching methods such as `redComponent` and `getRed:green:blue:alpha:` are only effective on `NSColor` objects based on named color spaces.

If you have an `NSColor` object in an unknown color space and you want to extract its components, you should first convert the color object to a known color space before using the component accessor methods of that color space.

Adopted Protocols

NSCoding
`encodeWithCoder:`
`initWithCoder:`

NSCopying
`copyWithZone:`

Tasks

Creating an NSColor Object from Component Values

`+ colorWithCalibratedHue:saturation:brightness:alpha:` (page 670)
Creates and returns an `NSColor` object using the given opacity and HSB color space components.

+ colorWithCalibratedRed:green:blue:alpha: (page 670)
Creates and returns an NSColor object using the given opacity and RGB components.

+ colorWithCalibratedWhite:alpha: (page 671)
Creates and returns an NSColor object using the given opacity and grayscale value.

+ colorWithCatalogName:colorName: (page 672)
Creates and returns an NSColor object by finding the color with the specified name in the given catalog.

+ colorWithDeviceCyan:magenta:yellow:black:alpha: (page 674)
Creates and returns an NSColor object using the given opacity value and CMYK components.

+ colorWithDeviceHue:saturation:brightness:alpha: (page 674)
Creates and returns an NSColor object using the given opacity value and HSB color space components.

+ colorWithDeviceRed:green:blue:alpha: (page 675)
Creates and returns an NSColor object using the given opacity value and RGB components.

+ colorWithDeviceWhite:alpha: (page 676)
Creates and returns an NSColor object using the given opacity and grayscale values.

+ colorWithCIColor: (page 672)
Converts a Core Image color object to its NSColor equivalent.

+ colorWithColorSpace:components:count: (page 673)
Returns an NSColor object created from the specified components of the given color space.

Creating an NSColor with Preset Components

+ blackColor (page 667)
Returns an NSColor object whose grayscale value is 0.0 and whose alpha value is 1.0.

+ blueColor (page 668)
Returns an NSColor object whose RGB value is 0.0, 0.0, 1.0 and whose alpha value is 1.0.

+ brownColor (page 668)
Returns an NSColor object whose RGB value is 0.6, 0.4, 0.2 and whose alpha value is 1.0.

+ clearColor (page 668)
Returns an NSColor object whose grayscale and alpha values are both 0.0.

+ cyanColor (page 681)
Returns an NSColor object whose RGB value is 0.0, 1.0, 1.0 and whose alpha value is 1.0.

+ darkGrayColor (page 681)
Returns an NSColor object whose grayscale value is 1/3 and whose alpha value is 1.0.

+ grayColor (page 682)
Returns an NSColor object whose grayscale value is 0.5 and whose alpha value is 1.0.

+ greenColor (page 682)
Returns an NSColor object whose RGB value is 0.0, 1.0, 0.0 and whose alpha value is 1.0.

+ lightGrayColor (page 686)
Returns an NSColor object whose grayscale value is 2/3 and whose alpha value is 1.0.

+ magentaColor (page 686)
Returns an NSColor object whose RGB value is 1.0, 0.0, 1.0 and whose alpha value is 1.0.

+ orangeColor (page 686)
Returns an NSColor object whose RGB value is 1.0, 0.5, 0.0 and whose alpha value is 1.0.

+ purpleColor (page 687)
    Returns an NSColor object whose RGB value is 0.5, 0.0, 0.5 and whose alpha value is 1.0.
+ redColor (page 687)
    Returns an NSColor object whose RGB value is 1.0, 0.0, 0.0 and whose alpha value is 1.0.
+ whiteColor (page 693)
    Returns an NSColor object whose grayscale and alpha values are both 1.0.
+ yellowColor (page 695)
    Returns an NSColor object whose RGB value is 1.0, 1.0, 0.0 and whose alpha value is 1.0.

Working with Pattern Images

+ colorWithPatternImage: (page 677)
    Creates and returns an NSColor object that uses the specified image pattern.
- patternImage (page 709)
    Returns the image that the receiver is using as a pattern.

Creating a System Color—an NSColor Whose Value Is Specified by User Preferences

+ alternateSelectedControlColor (page 666)
    Returns the system color used for the face of a selected control.
+ alternateSelectedControlTextColor (page 667)
    Returns the system color used for text in a selected control.
+ colorForControlTint: (page 669)
    Returns the NSColor object specified by the given control tint.
+ controlBackgroundColor (page 677)
    Returns the system color used for the background of large controls.
+ controlColor (page 678)
    Returns the system color used for the flat surfaces of a control.
+ controlAlternatingRowBackgroundColors (page 677)
    Returns an array containing the system specified background colors for alternating rows in tables and lists.
+ controlHighlightColor (page 679)
    Returns the system color used for the highlighted bezels of controls.
+ controlLightHighlightColor (page 679)
    Returns the system color used for light highlights in controls.
+ controlShadowColor (page 679)
    Returns the system color used for the shadows dropped from controls.
+ controlDarkShadowColor (page 678)
    Returns the system color used for the dark edge of the shadow dropped from controls.
+ controlTextColor (page 680)
    Returns the system color used for text on controls that aren't disabled.

+ currentControlTint (page 680)
  Returns the current system control tint.
+ disabledControlTextColor (page 681)
  Returns the system color used for text on disabled controls.
+ gridColor (page 683)
  Returns the system color used for the optional gridlines in, for example, a table view.
+ headerColor (page 683)
  Returns the system color used as the background color for header cells in table views and outline views.
+ headerTextColor (page 684)
  Returns the system color used for text in header cells in table views and outline views.
+ highlightColor (page 684)
  Returns the system color that represents the virtual light source on the screen.
+ keyboardFocusIndicatorColor (page 685)
  Returns the system color that represents the keyboard focus ring around controls.
+ knobColor (page 685)
  Returns the system color used for the flat surface of a slider knob that hasn't been selected.
+ scrollBarColor (page 688)
  Returns the system color used for scroll "bars"—that is, for the groove in which a scroller's knob moves
+ secondarySelectedControlColor (page 688)
  Returns the system color used in non-key views.
+ selectedControlColor (page 688)
  Returns the system color used for the face of a selected control.
+ selectedControlTextColor (page 689)
  Returns the system color used for text in a selected control—a control being clicked or dragged.
+ selectedMenuItemColor (page 690)
  Returns the system color used for the face of selected menu items.
+ selectedMenuItemTextColor (page 690)
  Returns the system color used for the text in menu items.
+ selectedTextBackgroundColor (page 690)
  Returns the system color used for the background of selected text.
+ selectedTextColor (page 691)
  Returns the system color used for selected text.
+ selectedKnobColor (page 689)
  Returns the system color used for the slider knob when it is selected.
+ shadowColor (page 692)
  Returns the system color that represents the virtual shadows cast by raised objects on the screen.
+ textBackgroundColor (page 692)
  Returns the system color used for the text background.
+ textColor (page 693)
  Returns the system color used for text.
+ windowBackgroundColor (page 694)
  Returns a pattern color that will draw the ruled lines for the window background.

+ windowFrameColor (page 694)
Returns the system color used for window frames, except for their text.

+ windowFrameTextColor (page 694)
Returns the system color used for the text in window frames.

Ignoring Alpha Components

+ ignoresAlpha (page 684)
Returns a Boolean value indicating whether the application supports alpha.

+ setIgnoresAlpha: (page 691)
Specifies whether an application supports alpha.

Copying and Pasting

+ colorFromPasteboard: (page 669)
Returns the NSColor currently on the given pasteboard.

- writeToPasteboard: (page 713)
Writes the receiver's data to the specified pasteboard.

Retrieving a Set of Components

- getCyan:magenta:yellow:black:alpha: (page 703)
Returns the receiver's CMYK and opacity values.

- getHue:saturation:brightness:alpha: (page 704)
Returns the receiver's HSB component and opacity values in the respective arguments.

- getRed:green:blue:alpha: (page 705)
Returns the receiver's RGB component and opacity values in the respective arguments.

- getWhite:alpha: (page 706)
Returns the receiver's grayscale value and alpha values.

- getComponents: (page 703)
Returns the components of the receiver as an array.

- numberOfComponents (page 709)
Returns the number of components in the receiver.

Retrieving Individual Components

- alphaComponent (page 695)
Returns the receiver's alpha (opacity) component.

- blackComponent (page 696)
Returns the receiver's black component.

- blueComponent (page 697)
Returns the receiver's blue component.

- `brightnessComponent` (page 697)
  Returns the brightness component of the HSB color equivalent to the receiver.
- `catalogNameComponent` (page 698)
  Returns the name of the catalog containing the receiver's name.
- `colorNameComponent` (page 698)
  Returns the receiver's name.
- `cyanComponent` (page 702)
  Returns the receiver's cyan component.
- `greenComponent` (page 706)
  Returns the receiver's green component.
- `hueComponent` (page 707)
  Returns the hue component of the HSB color equivalent to the receiver.
- `localizedCatalogNameComponent` (page 708)
  Returns the name of the catalog containing the receiver's name as a localized string.
- `localizedColorNameComponent` (page 708)
  Returns the name of the receiver as a localized string.
- `magentaComponent` (page 708)
  Returns the receiver's magenta component.
- `redComponent` (page 710)
  Returns the receiver's red component.
- `saturationComponent` (page 710)
  Returns the saturation component of the HSB color equivalent to the receiver.
- `whiteComponent` (page 712)
  Returns the receiver's white component.
- `yellowComponent` (page 713)
  Returns the receiver's yellow component.

Working with the Color Space

- `colorSpaceName` (page 699)
  Returns the name of the receiver's color space.
- `colorUsingColorSpaceName:` (page 700)
  Creates and returns an `NSColor` whose color is the same as the receiver's, except that the new `NSColor` is in the specified color space.
- `colorUsingColorSpaceName:device:` (page 701)
  Creates and returns an `NSColor` object whose color is the same as the receiver's, except that the new NSColor is in the given color space and is specific to the given device.
- `colorSpace` (page 699)
  Returns an object representing the color space of the receiver.
- `colorUsingColorSpace:` (page 700)
  Returns a new color object representing the color of the receiver in the specified color space.

Changing the Color

- `blendedColorWithFraction:ofColor:` (page 696)
  Creates and returns an `NSColor` object whose component values are a weighted sum of the receiver's and the specified color object's.
- `colorWithAlphaComponent:` (page 701)
  Creates and returns an `NSColor` object that has the same color space and component values as the receiver, but the specified alpha component.
- `highlightWithLevel:` (page 707)
  Returns an `NSColor` object that represents a blend between the receiver and the highlight color returned by `highlightColor` (page 684).
- `shadowWithLevel:` (page 712)
  Returns an `NSColor` object that represents a blend between the receiver and the shadow color returned by `shadowColor` (page 692).

Drawing

- `drawSwatchInRect:` (page 702)
  Draws the current color in the given rectangle.
- `set` (page 711)
  Sets the color of subsequent drawing to the color that the receiver represents.
- `setFill` (page 711)
  Sets the fill color of subsequent drawing to the receiver's color.
- `setStroke` (page 711)
  Sets the stroke color of subsequent drawing to the receiver's color.

Class Methods

alternateSelectedControlColor

Returns the system color used for the face of a selected control.

```
+ (NSColor *)alternateSelectedControlColor
```

Return Value
The system color used for the face of a selected control—a control being clicked or dragged. This color can be used where iApp-like highlighting is desired. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ alternateSelectedControlTextColor` (page 667)
`+ selectedControlColor` (page 688)

Declared In
NSColor.h

alternateSelectedControlTextColor

Returns the system color used for text in a selected control.

```
+ (NSColor *)alternateSelectedControlTextColor
```

Return Value
The system color used for text in a selected control—a control being clicked or dragged. This color can be used where iApp-like highlighting is desired. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.2 and later.

See Also
+ alternateSelectedControlColor (page 666)
+ selectedControlTextColor (page 689)

Related Sample Code
PhotoSearch

Declared In
NSColor.h

blackColor

Returns an NSColor object whose grayscale value is 0.0 and whose alpha value is 1.0.

```
+ (NSColor *)blackColor
```

Return Value
The NSColor object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- blackComponent (page 696)

Related Sample Code
DockTile
PDF Annotation Editor
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSColor.h

blueColor

Returns an `NSColor` object whose RGB value is 0.0, 0.0, 1.0 and whose alpha value is 1.0.

```
+ (NSColor *)blueColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- blueComponent` (page 697)

Related Sample Code
FunkyOverlayWindow
Grady
VertexPerformanceTest
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSColor.h`

brownColor

Returns an `NSColor` object whose RGB value is 0.6, 0.4, 0.2 and whose alpha value is 1.0.

```
+ (NSColor *)brownColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
StickiesWithCoreData

Declared In
`NSColor.h`

clearColor

Returns an `NSColor` object whose grayscale and alpha values are both 0.0.

```
+ (NSColor *)clearColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AnimatedTableView
CompositeLab
FunkyOverlayWindow
Sketch-112
TrackBall

Declared In
NSColor.h

colorForControlTint:

Returns the NSColor object specified by the given control tint.

```
+ (NSColor *)colorForControlTint:(NSControlTint)controlTint
```

Parameters
controlTint
The control tint for which to return an NSColor object. This is one of the tint settings. For more on control tints, see Using the System Control Tint.

Return Value
The NSColor object.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ currentControlTint (page 680)

Declared In
NSColor.h

colorFromPasteboard:

Returns the NSColor currently on the given pasteboard.

```
+ (NSColor *)colorFromPasteboard:(NSPasteboard *)pasteBoard
```

Parameters
pasteBoard
The pasteboard from which to return the color.

Return Value
The color currently on the pasteboard or nil if *pasteBoard* doesn't contain color data. The returned color's alpha component is set to 1.0 if ignoresAlpha (page 684) returns YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- writeToPasteboard: (page 713)

Related Sample Code
CompositeLab
Sketch+Accessibility
Sketch-112

Declared In
NSColor.h

colorWithCalibratedHue:saturation:brightness:alpha:

Creates and returns an NSColor object using the given opacity and HSB color space components.

```
+ (NSColor *)colorWithCalibratedHue:(CGFloat)hue saturation:(CGFloat)saturation
    brightness:(CGFloat)brightness alpha:(CGFloat)alpha
```

Parameters

hue
The hue component of the color object in the HSB color space.

saturation
The saturation component of the color object in the HSB color space.

brightness
The brightness (or value) component of the color object in the HSB color space.

alpha
The opacity value of the color object,

Return Value
The color object.

Discussion
Values below 0.0 are interpreted as 0.0, and values above 1.0 are interpreted as 1.0.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ colorWithCalibratedRed:green:blue:alpha: (page 670)
+ colorWithDeviceHue:saturation:brightness:alpha: (page 674)
- getHue:saturation:brightness:alpha: (page 704)

Related Sample Code
NewsReader

Declared In
NSColor.h

colorWithCalibratedRed:green:blue:alpha:

Creates and returns an NSColor object using the given opacity and RGB components.

```
+ (NSColor *)colorWithCalibratedRed:(CGFloat)red green:(CGFloat)green
    blue:(CGFloat)blue alpha:(CGFloat)alpha
```

Parameters

red

The red component of the color object.

green

The green component of the color object.

blue

The blue component of the color object.

alpha

The opacity value of the color object.

Return Value
The color object.

Discussion
Values below 0.0 are interpreted as 0.0, and values above 1.0 are interpreted as 1.0.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ colorWithCalibratedHue:saturation:brightness:alpha:` (page 670)
`+ colorWithDeviceRed:green:blue:alpha:` (page 675)
`- getRed:green:blue:alpha:` (page 705)

Related Sample Code
Color Sampler
ImageApp
ImageKitDemo
SonogramViewDemo
WhackedTV

Declared In
`NSColor.h`

colorWithCalibratedWhite:alpha:

Creates and returns an `NSColor` object using the given opacity and grayscale value.

```
+ (NSColor *)colorWithCalibratedWhite:(CGFloat)white alpha:(CGFloat)alpha
```

Parameters

white

The grayscale value of the color object.

alpha

The opacity value of the color object.

Return Value
The color object.

Discussion
Values below 0.0 are interpreted as 0.0, and values above 1.0 are interpreted as 1.0.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ colorWithDeviceWhite:alpha:` (page 676)
`- getWhite:alpha:` (page 706)

Related Sample Code
ClockControl
CocoaSlides
MatrixMixerTest
SampleRaster
WhackedTV

Declared In
`NSColor.h`

colorWithCatalogName:colorName:

Creates and returns an `NSColor` object by finding the color with the specified name in the given catalog.

```
+ (NSColor *)colorWithCatalogName:(NSString *)listName colorName:(NSString
    *)colorName
```

Parameters

`listName`
The name of the catalog in which to find the specified color; this may be a standard catalog.

`colorName`
The name of the color. Note that the color must be defined in the named color space to retrieve it with this method.

Return Value
The color object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- catalogNameComponent` (page 698)
`- colorNameComponent` (page 698)
`- localizedCatalogNameComponent` (page 708)

Declared In
`NSColor.h`

colorWithCIColor:

Converts a Core Image color object to its `NSColor` equivalent.

```
+ (NSColor *)colorWithCIColor:(CIColor *)color
```

Parameters

color

The Core Image color to convert.

Return Value

The `NSColor` object corresponding to the specified Core Image color.

Discussion

The method raises if the color space and components associated with *color* are `nil` or invalid.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

CIColorTracking

FunHouse

Declared In

`NSColor.h`

colorWithColorSpace:components:count:

Returns an `NSColor` object created from the specified components of the given color space.

```
+ (NSColor *)colorWithColorSpace:(NSColorSpace *)space components:(const CGFloat
    *)components count:(NSInteger)numberOfComponents
```

Parameters

space

An `NSColorSpace` object representing a color space. The method raises if this is `nil`.

components

An array of the components in the specified color space to use to create the `NSColor` object. The order of these components is determined by the color-space profile, with the alpha component always last. (If you want the created color to be opaque, specify 1.0 for the alpha component.)

numberOfComponents

The number of components in the `components` array. This should match the number dictated by the specified color space plus one for alpha. This method raises an exception if they do not match.

Return Value

The color object. If *space* represents a color space that cannot cannot be used with `NSColor` objects—for example, a "pattern" color space—the method returns `nil`.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- colorUsingColorSpace:` (page 700)

Related Sample Code

AnimatedTableView

Declared In
NSColor.h

colorWithDeviceCyan:magenta:yellow:black:alpha:

Creates and returns an NSColor object using the given opacity value and CMYK components.

```
+ (NSColor *)colorWithDeviceCyan:(CGFloat)cyan magenta:(CGFloat)magenta
    yellow:(CGFloat)yellow black:(CGFloat)black alpha:(CGFloat)alpha
```

Parameters

cyan
The cyan component of the color object.

magenta
The magenta component of the color object.

yellow
The yellow component of the color object.

black
The black component of the color object.

alpha
The opacity value of the color object.

Return Value
The color object.

Discussion
Values below 0.0 are interpreted as 0.0, and values above 1.0 are interpreted as 1.0. In PostScript, this color space corresponds directly to the device-dependent operator setcmykcolor.

Availability
Available in Mac OS X v10.0 and later.

See Also
- getCyan:magenta:yellow:black:alpha: (page 703)

Declared In
NSColor.h

colorWithDeviceHue:saturation:brightness:alpha:

Creates and returns an NSColor object using the given opacity value and HSB color space components.

```
+ (NSColor *)colorWithDeviceHue:(CGFloat)hue saturation:(CGFloat)saturation
    brightness:(CGFloat)brightness alpha:(CGFloat)alpha
```

Parameters

hue
The hue component of the color object.

saturation
The saturation component of the color object.

brightness
The brightness component of the color object.

alpha
The opacity value of the color object.

Return Value
The color object.

Discussion
Values below 0.0 are interpreted as 0.0, and values above 1.0 are interpreted as 1.0. In PostScript, this color space corresponds directly to the device-dependent operator `setrgbcolor`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ colorWithCalibratedHue:saturation:brightness:alpha:` (page 670)
`+ colorWithDeviceRed:green:blue:alpha:` (page 675)
`- getHue:saturation:brightness:alpha:` (page 704)

Related Sample Code
MenuItemView

Declared In
`NSColor.h`

colorWithDeviceRed:green:blue:alpha:

Creates and returns an `NSColor` object using the given opacity value and RGB components.

```
+ (NSColor *)colorWithDeviceRed:(CGFloat)red green:(CGFloat)green blue:(CGFloat)blue
     alpha:(CGFloat)alpha
```

Parameters

red
The red component of the color object.

green
The green component of the color object.

blue
The blue component of the color object.

alpha
The opacity value of the color object.

Return Value
The color object.

Discussion
Values below 0.0 are interpreted as 0.0, and values above 1.0 are interpreted as 1.0. In PostScript, this color space corresponds directly to the device-dependent operator `setrgbcolor`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ colorWithCalibratedRed:green:blue:alpha:` (page 670)
`+ colorWithDeviceHue:saturation:brightness:alpha:` (page 674)
`- getRed:green:blue:alpha:` (page 705)

Related Sample Code
Cocoa OpenGL
From A View to A Movie
From A View to A Picture
IconCollection
SourceView

Declared In
`NSColor.h`

colorWithDeviceWhite:alpha:

Creates and returns an `NSColor` object using the given opacity and grayscale values.

```
+ (NSColor *)colorWithDeviceWhite:(CGFloat)white alpha:(CGFloat)alpha
```

Parameters

white
The grayscale value of the color object.

alpha
The opacity value of the color object.

Return Value
The color object.

Discussion
Values below 0.0 are interpreted as 0.0, and values above 1.0 are interpreted as 1.0. In PostScript, this color space corresponds directly to the device-dependent operator `setgray`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ colorWithCalibratedWhite:alpha:` (page 671)
`- getWhite:alpha:` (page 706)

Related Sample Code
FilterDemo
Link Snoop
PDFKitLinker2

Declared In
`NSColor.h`

colorWithPatternImage:

Creates and returns an `NSColor` object that uses the specified image pattern.

```
+ (NSColor *)colorWithPatternImage:(NSImage *)image
```

Parameters

image

The image to use as the pattern for the color object. The image is tiled starting at the bottom of the window. The image is not scaled.

Return Value
The `NSColor` object. This color object is autoreleased.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Dicey
FilterDemo
Worm

Declared In
`NSColor.h`

controlAlternatingRowBackgroundColors

Returns an array containing the system specified background colors for alternating rows in tables and lists.

```
+ (NSArray *)controlAlternatingRowBackgroundColors
```

Return Value
An array of `NSColor` objects specifying the system colors used for rows in tables and lists. You should not assume the array will contain only two colors. For general information on system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSColor.h`

controlBackgroundColor

Returns the system color used for the background of large controls.

```
+ (NSColor *)controlBackgroundColor
```

Return Value
The system color used for the background of large controls such as browsers, table views, and clip views. For general information on system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSColor.h

controlColor

Returns the system color used for the flat surfaces of a control.

```
+ (NSColor *)controlColor
```

Return Value
The system color used for the flat surfaces of a control. By default, the control color is a pattern color that will draw the ruled lines for the window background, which is the same as returned by `windowBackgroundColor` (page 694).

If you use `controlColor` assuming that it is a solid, you may have an incorrect appearance. You should use `lightGrayColor` (page 686) in its place.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSColor.h

controlDarkShadowColor

Returns the system color used for the dark edge of the shadow dropped from controls.

```
+ (NSColor *)controlDarkShadowColor
```

Return Value
Of the two dark borders that run along the bottom and right of controls, representing shadows, the color of the outer, darker border. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ controlShadowColor` (page 679)

Related Sample Code
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
NSColor.h

controlHighlightColor

Returns the system color used for the highlighted bezels of controls.

```
+ (NSColor *)controlHighlightColor
```

Return Value
Of the two light borders that run along the top and left of controls, representing reflections from a light source in the upper left, the color of the inner, duller border. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ controlLightHighlightColor (page 679)

Related Sample Code
DictionaryController

Declared In
NSColor.h

controlLightHighlightColor

Returns the system color used for light highlights in controls.

```
+ (NSColor *)controlLightHighlightColor
```

Return Value
Of the two light borders that run along the top and left of controls, representing reflections from a light source in the upper left, the color of the outer, brighter border. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ controlHighlightColor (page 679)

Declared In
NSColor.h

controlShadowColor

Returns the system color used for the shadows dropped from controls.

```
+ (NSColor *)controlShadowColor
```

Return Value
Of the two dark borders that run along the bottom and right of controls, representing shadows, the color of the inner, lighter border. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ controlDarkShadowColor` (page 678)

Declared In
`NSColor.h`

controlTextColor

Returns the system color used for text on controls that aren't disabled.

```
+ (NSColor *)controlTextColor
```

Return Value
The color used for text on enabled controls. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ disabledControlTextColor` (page 681)

Related Sample Code
NewsReader
TextSizingExample

Declared In
`NSColor.h`

currentControlTint

Returns the current system control tint.

```
+ (NSControlTint)currentControlTint
```

Return Value
The current system control tint.

Discussion
An application can register for the `NSControlTintDidChangeNotification` (page 624) notification to be notified of changes to the system control tint.

Availability
Available in Mac OS X v10.3 and later.

See Also
`+ colorForControlTint:` (page 669)

Declared In
`NSColor.h`

cyanColor

Returns an `NSColor` object whose RGB value is 0.0, 1.0, 1.0 and whose alpha value is 1.0.

```
+ (NSColor *)cyanColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- cyanComponent` (page 702)

Related Sample Code
LayoutManagerDemo
RGB Image

Declared In
`NSColor.h`

darkGrayColor

Returns an `NSColor` object whose grayscale value is 1/3 and whose alpha value is 1.0.

```
+ (NSColor *)darkGrayColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ lightGrayColor` (page 686)
`+ grayColor` (page 682)

Related Sample Code
iChatTheater
QuickLookDownloader
TextLinks
TrackIt

Declared In
`NSColor.h`

disabledControlTextColor

Returns the system color used for text on disabled controls.

```
+ (NSColor *)disabledControlTextColor
```

Return Value
The color used for text on disabled controls. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ controlTextColor` (page 680)

Related Sample Code
NewsReader

Declared In
`NSColor.h`

grayColor

Returns an `NSColor` object whose grayscale value is 0.5 and whose alpha value is 1.0.

```
+ (NSColor *)grayColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ lightGrayColor` (page 686)
`+ darkGrayColor` (page 681)

Related Sample Code
ImageMap
ImageMapExample
PhotoSearch
UIElementInspector
URL CacheInfo

Declared In
`NSColor.h`

greenColor

Returns an `NSColor` object whose RGB value is 0.0, 1.0, 0.0 and whose alpha value is 1.0.

```
+ (NSColor *)greenColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- greenComponent (page 706)

Related Sample Code
MatrixMixerTest
TargetGallery
WebKitPluginStarter
WebKitPluginWithJavaScript
WhackedTV

Declared In
NSColor.h

gridColor

Returns the system color used for the optional gridlines in, for example, a table view.

```
+ (NSColor *)gridColor
```

Return Value
The system color used for gridlines. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
IconCollection

Declared In
NSColor.h

headerColor

Returns the system color used as the background color for header cells in table views and outline views.

```
+ (NSColor *)headerColor
```

Return Value
The system color used as the background for header cells in table and outline views. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSColor.h

headerTextColor

Returns the system color used for text in header cells in table views and outline views.

```
+ (NSColor *)headerTextColor
```

Return Value
The system color used for text in header cells in table and outline views. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSColor.h

highlightColor

Returns the system color that represents the virtual light source on the screen.

```
+ (NSColor *)highlightColor
```

Return Value
The system color for the virtual light source on the screen.

Discussion
This method is invoked by the `highlightWithLevel:` (page 707) method. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- highlightWithLevel:` (page 707)

Declared In
NSColor.h

ignoresAlpha

Returns a Boolean value indicating whether the application supports alpha.

```
+ (BOOL)ignoresAlpha
```

Return Value
`YES` if the application doesn't support alpha; otherwise `NO`. This value is consulted when an application imports alpha (through color dragging, for instance). The value determines whether the color panel has an opacity slider.

This value is `YES` by default, indicating that the opacity components of imported colors will be set to 1.0. If an application wants alpha, it can invoke the `setIgnoresAlpha:` (page 691) method with a parameter of `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ setIgnoresAlpha:` (page 691)
`- alphaComponent` (page 695)

Declared In
`NSColor.h`

keyboardFocusIndicatorColor

Returns the system color that represents the keyboard focus ring around controls.

```
+ (NSColor *)keyboardFocusIndicatorColor
```

Return Value
The system color representing the focus ring.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColor.h`

knobColor

Returns the system color used for the flat surface of a slider knob that hasn't been selected.

```
+ (NSColor *)knobColor
```

Return Value
The system color used for an unselected slider knob.

Discussion
The knob's beveled edges, which set it in relief, are drawn in highlighted and shadowed versions of the face color. When a knob is selected, its color changes to `selectedKnobColor` (page 689). For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSColor.h`

lightGrayColor

Returns an `NSColor` object whose grayscale value is 2/3 and whose alpha value is 1.0.

```
+ (NSColor *)lightGrayColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ grayColor` (page 682)
`+ darkGrayColor` (page 681)

Related Sample Code
IconCollection
IdentitySample
Quartz Composer WWDC 2005 TextEdit
Sketch-112
TrackIt

Declared In
`NSColor.h`

magentaColor

Returns an `NSColor` object whose RGB value is 1.0, 0.0, 1.0 and whose alpha value is 1.0.

```
+ (NSColor *)magentaColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- magentaComponent` (page 708)

Related Sample Code
CocoaVideoFrameToGWorld
LayoutManagerDemo
RGB Image

Declared In
`NSColor.h`

orangeColor

Returns an `NSColor` object whose RGB value is 1.0, 0.5, 0.0 and whose alpha value is 1.0.

```
+ (NSColor *)orangeColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DispatchFractal
Grady
MenuItemView

Declared In
`NSColor.h`

purpleColor

Returns an `NSColor` object whose RGB value is 0.5, 0.0, 0.5 and whose alpha value is 1.0.

```
+ (NSColor *)purpleColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MenuItemView

Declared In
`NSColor.h`

redColor

Returns an `NSColor` object whose RGB value is 1.0, 0.0, 0.0 and whose alpha value is 1.0.

```
+ (NSColor *)redColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- redComponent` (page 710)

Related Sample Code
MatrixMixerTest
UIElementInspector
WebKitPluginStarter

WebKitPluginWithJavaScript
WhackedTV

Declared In
NSColor.h

scrollBarColor

Returns the system color used for scroll "bars"—that is, for the groove in which a scroller's knob moves

```
+ (NSColor *)scrollBarColor
```

Return Value
The system color used for scroll bars. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSColor.h

secondarySelectedControlColor

Returns the system color used in non-key views.

```
+ (NSColor *)secondarySelectedControlColor
```

Return Value
The system color used in non-key views. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.1 and later.

See Also
+ selectedControlColor (page 688)

Declared In
NSColor.h

selectedControlColor

Returns the system color used for the face of a selected control.

```
+ (NSColor *)selectedControlColor
```

Return Value
The system color used for the face of a selected control—a control being dragged or clicked. For general information about system colors, see Accessing System Colors

Availability
Available in Mac OS X v10.0 and later.

See Also
+ selectedControlTextColor (page 689)
+ secondarySelectedControlColor (page 688)
+ alternateSelectedControlColor (page 666)

Related Sample Code
CocoaAUHost
DictionaryController

Declared In
NSColor.h

selectedControlTextColor

Returns the system color used for text in a selected control—a control being clicked or dragged.

```
+ (NSColor *)selectedControlTextColor
```

Return Value
The system color used for text in a selected control—a control being clicked or dragged. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ selectedControlColor (page 688)
+ alternateSelectedControlTextColor (page 667)

Related Sample Code
TextSizingExample

Declared In
NSColor.h

selectedKnobColor

Returns the system color used for the slider knob when it is selected.

```
+ (NSColor *)selectedKnobColor
```

Return Value
The system color used for a slider knob that is selected—that is, dragged. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ knobColor (page 685)

Declared In
NSColor.h

selectedMenuItemColor

Returns the system color used for the face of selected menu items.

```
+ (NSColor *)selectedMenuItemColor
```

Return Value
The system color used for selected menu items. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ selectedMenuItemTextColor (page 690)

Declared In
NSColor.h

selectedMenuItemTextColor

Returns the system color used for the text in menu items.

```
+ (NSColor *)selectedMenuItemTextColor
```

Return Value
The system color used for text in selected menu items. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ selectedMenuItemColor (page 690)

Declared In
NSColor.h

selectedTextBackgroundColor

Returns the system color used for the background of selected text.

```
+ (NSColor *)selectedTextBackgroundColor
```

Return Value
The system color used for the background of selected text. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ selectedTextColor` (page 691)

Related Sample Code
TextSizingExample

Declared In
`NSColor.h`

selectedTextColor

Returns the system color used for selected text.

```
+ (NSColor *)selectedTextColor
```

Return Value
The system color used for selected text. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ selectedTextBackgroundColor` (page 690)

Declared In
`NSColor.h`

setIgnoresAlpha:

Specifies whether an application supports alpha.

```
+ (void)setIgnoresAlpha:(BOOL)flag
```

Parameters

flag

`YES` to indicate that the application won't support alpha. By default, applications ignore alpha.

Discussion
If the application doesn't support alpha, no opacity slider is displayed in the color panel, and colors dragged in or pasted have their alpha values set to 1.0. Applications that need to import alpha can invoke this method with *flag* set to `NO` and explicitly make colors opaque in cases where it matters to them. Note that calling this with a value of `YES` overrides any value set with the `NSColorPanel` method `setShowsAlpha:` (page 735).

Availability
Available in Mac OS X v10.0 and later.

See Also
+ ignoresAlpha (page 684)
- alphaComponent (page 695)

Related Sample Code
Aperture Edit Plugin - Borders & Titles
CompositeLab
Quartz Composer QCTV
Tinted Image

Declared In
NSColor.h

shadowColor

Returns the system color that represents the virtual shadows cast by raised objects on the screen.

```
+ (NSColor *)shadowColor
```

Return Value
The system color for the virtual shadows case by raised objects on the screen.

Discussion
This method is invoked by shadowWithLevel: (page 712). For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
- shadowWithLevel: (page 712)

Declared In
NSColor.h

textBackgroundColor

Returns the system color used for the text background.

```
+ (NSColor *)textBackgroundColor
```

Return Value
The system color used for the background of text. When text is selected, its background color changes to the return value of selectedTextBackgroundColor (page 690). For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ textColor (page 693)

Related Sample Code
TextSizingExample

Declared In
NSColor.h

textColor

Returns the system color used for text.

```
+ (NSColor *)textColor
```

Return Value
The system color used for text. When text is selected, its color changes to the return value of `selectedTextColor` (page 691). For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ textBackgroundColor` (page 692)

Declared In
NSColor.h

whiteColor

Returns an `NSColor` object whose grayscale and alpha values are both 1.0.

```
+ (NSColor *)whiteColor
```

Return Value
The `NSColor` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- whiteComponent` (page 712)

Related Sample Code
Cocoa OpenGL
DragItemAround
FilterDemo
Quartz Composer WWDC 2005 TextEdit
Sketch-112

Declared In
NSColor.h

windowBackgroundColor

Returns a pattern color that will draw the ruled lines for the window background.

```
+ (NSColor *)windowBackgroundColor
```

Return Value
The pattern color used for the background of a window. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AnimatedTableView

Declared In
NSColor.h

windowFrameColor

Returns the system color used for window frames, except for their text.

```
+ (NSColor *)windowFrameColor
```

Return Value
The system color used for window frames. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ windowFrameTextColor (page 694)

Declared In
NSColor.h

windowFrameTextColor

Returns the system color used for the text in window frames.

```
+ (NSColor *)windowFrameTextColor
```

Return Value
The system color used for text in window frames. For general information about system colors, see Accessing System Colors.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ windowFrameColor (page 694)

Declared In
NSColor.h

yellowColor

Returns an NSColor object whose RGB value is 1.0, 1.0, 0.0 and whose alpha value is 1.0.

```
+ (NSColor *)yellowColor
```

Return Value
The NSColor object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- yellowComponent (page 713)

Related Sample Code
LayoutManagerDemo
RGB Image
SimpleStickies
StickiesWithCoreData
WebKitPluginStarter

Declared In
NSColor.h

Instance Methods

alphaComponent

Returns the receiver's alpha (opacity) component.

```
- (CGFloat)alphaComponent
```

Return Value
The alpha component of the color object. If the receiver has no alpha component, this is 1.0 (opaque).

Availability
Available in Mac OS X v10.0 and later.

See Also
- getCyan:magenta:yellow:black:alpha: (page 703)
- getHue:saturation:brightness:alpha: (page 704)
- getRed:green:blue:alpha: (page 705)
- getWhite:alpha: (page 706)

Related Sample Code
DragItemAround

Monochrome Image

Declared In
NSColor.h

blackComponent

Returns the receiver's black component.

```
- (CGFloat)blackComponent
```

Return Value
The color object's black component.

Discussion
This method works only with objects representing colors in the `NSDeviceCMYKColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- getCyan:magenta:yellow:black:alpha:` (page 703)

Declared In
NSColor.h

blendedColorWithFraction:ofColor:

Creates and returns an `NSColor` object whose component values are a weighted sum of the receiver's and the specified color object's.

```
- (NSColor *)blendedColorWithFraction:(CGFloat)fraction ofColor:(NSColor *)color
```

Parameters

fraction
The amount of the color to blend with the receiver's color. The method converts *color* and a copy of the receiver to RGB, and then sets each component of the returned color to *fraction* of *color*'s value plus 1 – *fraction* of the receiver's.

color
The color to blend with the receiver's color.

Return Value
The resulting color object or `nil` if the colors can't be converted.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DockTile
SpeedometerView
WebKitPluginStarter

WebKitPluginWithJavaScript

Declared In
NSColor.h

blueComponent

Returns the receiver's blue component.

```
- (CGFloat)blueComponent
```

Return Value
The color object's blue component.

Discussion
This method works only with objects representing colors in the `NSCalibratedRGBColorSpace` or `NSDeviceRGBColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- getRed:green:blue:alpha:` (page 705)

Related Sample Code
CIColorTracking
Color Sampler
QTCoreVideo301
Sketch+Accessibility

Declared In
NSColor.h

brightnessComponent

Returns the brightness component of the HSB color equivalent to the receiver.

```
- (CGFloat)brightnessComponent
```

Return Value
The color object's brightness component.

Discussion
This method works only with objects representing colors in the `NSCalibratedRGBColorSpace` or `NSDeviceRGBColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- getHue:saturation:brightness:alpha:` (page 704)

Declared In
NSColor.h

catalogNameComponent

Returns the name of the catalog containing the receiver's name.

```
- (NSString *)catalogNameComponent
```

Return Value
The name of the catalog containing the color object.

Discussion
This method raises an exception if the receiver's color space isn't `NSNamedColorSpace`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ colorWithCatalogName:colorName:` (page 672)
`- colorNameComponent` (page 698)
`- localizedCatalogNameComponent` (page 708)

Declared In
NSColor.h

colorNameComponent

Returns the receiver's name.

```
- (NSString *)colorNameComponent
```

Return Value
The name of the color object.

Discussion
This method raises an exception if the receiver's color space isn't `NSNamedColorSpace`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ colorWithCatalogName:colorName:` (page 672)
`- catalogNameComponent` (page 698)
`- localizedCatalogNameComponent` (page 708)

Declared In
NSColor.h

colorSpace

Returns an object representing the color space of the receiver.

```
- (NSColorSpace *)colorSpace
```

Return Value
An object representing a color space. The returned `NSColorSpace` object may represent a custom color space.

Discussion
Calling this method raises an exception if the receiver is not based on a color space represented by an `NSColorSpace` object—specifically, colors designated by `NSNamedColorSpace` and `NSPatternColorSpace`. If you are unsure about a color object, convert it to an equivalent `NSColorSpace`-based object before calling this method. Color objects created with color-space names `NSCalibratedWhiteColorSpace`, `NSCalibratedBlackColorSpace`, `NSCalibratedRGBColorSpace`, `NSDeviceWhiteColorSpace`, `NSDeviceBlackColorSpace`, `NSDeviceRGBColorSpace`, `NSDeviceCMYKColorSpace`, or `NSCustomColorSpace`—or with the `NSColorSpace` class methods corresponding to these names—are safe to use with this method. See “About Color Spaces" in *Color Programming Topics* for a list of these corresponding methods.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `getComponents:` (page 703)
- `numberOfComponents` (page 709)

Declared In
`NSColor.h`

colorSpaceName

Returns the name of the receiver’s color space.

```
- (NSString *)colorSpaceName
```

Return Value
The name of the color space.

Discussion
This method should be implemented in subclasses of `NSColor`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `colorUsingColorSpaceName:` (page 700)
- `colorUsingColorSpaceName:device:` (page 701)

Related Sample Code
MyCustomColorPicker

Declared In
NSColor.h

colorUsingColorSpace:

Returns a new color object representing the color of the receiver in the specified color space.

```
- (NSColor *)colorUsingColorSpace:(NSColorSpace *)space
```

Parameters

space

The color space of the new NSColor object.

Return Value
The new NSColor object. This method converts the receiver's color to an equivalent one in the new color space. Although the new color might have different component values, it looks the same as the original. Returns nil if conversion is not possible.

If the receiver's color space is the same as that specified in space, this method returns the same NSColor object.

Availability
Available in Mac OS X v10.4 and later.

See Also
+ colorWithColorSpace:components:count: (page 673)

Related Sample Code
Quartz 2D Shadings
Quartz Composer QCTV

Declared In
NSColor.h

colorUsingColorSpaceName:

Creates and returns an NSColor whose color is the same as the receiver's, except that the new NSColor is in the specified color space.

```
- (NSColor *)colorUsingColorSpaceName:(NSString *)colorSpace
```

Parameters

colorSpace

The name of the color space containing the new NSColor object. If *colorSpace* is nil, the most appropriate color space is used.

Return Value
The new NSColor object or nil if the specified conversion cannot be done.

Availability
Available in Mac OS X v10.0 and later.

See Also
- colorSpaceName (page 699)

Related Sample Code
CoreAnimationText
CWCocoaComponent
DispatchFractal
MenuItemView
Reminders

Declared In
NSColor.h

colorUsingColorSpaceName:device:

Creates and returns an NSColor object whose color is the same as the receiver's, except that the new NSColor is in the given color space and is specific to the given device.

```
- (NSColor *)colorUsingColorSpaceName:(NSString *)colorSpace device:(NSDictionary
    *)deviceDescription
```

Parameters

colorSpace

The name of the color space containing the new NSColor object. If *colorSpace* is nil, the most appropriate color space is used.

deviceDescription

The device description. Device descriptions can be obtained from windows, screens, and printers with the deviceDescription method.

If *deviceDescription* is nil, the current device (as obtained from the currently lockFocus'ed view's window or, if printing, the current printer) is used.

Return Value
The new NSColor object or nil if the specified conversion cannot be done.

Availability
Available in Mac OS X v10.0 and later.

See Also
- colorSpaceName (page 699)
- colorUsingColorSpaceName: (page 700)

Declared In
NSColor.h

colorWithAlphaComponent:

Creates and returns an NSColor object that has the same color space and component values as the receiver, but the specified alpha component.

```
- (NSColor *)colorWithAlphaComponent:(CGFloat)alpha
```

Parameters

alpha

The opacity value of the new NSColor object.

Return Value
The new `NSColor` object. If the receiver's color space doesn't include an alpha component, the receiver is returned.

Discussion
A subclass with explicit opacity components should override this method to return a color with the specified alpha.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alphaComponent` (page 695)
- `blendedColorWithFraction:ofColor:` (page 696)

Related Sample Code
FunkyOverlayWindow
ImageMap
ImageMapExample
Sketch-112
WebKitPluginStarter

Declared In
`NSColor.h`

cyanComponent

Returns the receiver's cyan component.

```
- (CGFloat)cyanComponent
```

Return Value
The color object's cyan component.

Discussion
This method works only with objects representing colors in the `NSDeviceCMYKColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `getCyan:magenta:yellow:black:alpha:` (page 703)

Declared In
`NSColor.h`

drawSwatchInRect:

Draws the current color in the given rectangle.

```
- (void)drawSwatchInRect:(NSRect)rect
```

Parameters

rect

The rectangle in which to draw the color.

Discussion

Subclasses adorn the rectangle in some manner to indicate the type of color. This method is invoked by color wells, swatches, and other user interface objects that need to display colors.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSColor.h

getComponents:

Returns the components of the receiver as an array.

```
- (void)getComponents:(CGFloat *)components
```

Parameters

components

An array containing the components of the color object as `float` values.

Discussion

You can invoke this method on `NSColor` objects created from custom color spaces to get the individual floating point components, including alpha. Raises an exception if the receiver doesn't have floating-point components. To find out how many components are in the *components* array, send the receiver a `numberOfComponents` (page 709) message.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- colorSpace` (page 699)

Related Sample Code

AnimatedTableView

CoreAnimationText

DispatchFractal

Quartz Composer QCTV

Declared In

NSColor.h

getCyan:magenta:yellow:black:alpha:

Returns the receiver's CMYK and opacity values.

```
- (void)getCyan:(CGFloat *)cyan magenta:(CGFloat *)magenta yellow:(CGFloat *)yellow
    black:(CGFloat *)black alpha:(CGFloat *)alpha
```

Parameters

cyan

Upon return, contains the cyan component of the color object.

magenta

Upon return, contains the magenta component of the color object.

yellow

Upon return, contains the yellow component of the color object.

black

Upon return, contains the black component of the color object.

alpha

Upon return, contains opacity value of the color object.

Discussion

If `NULL` is passed in as an argument, the method doesn't set that value. This method works only with objects representing colors in the `NSDeviceCMYKColorSpace`. Sending it to other objects raises an exception.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `alphaComponent` (page 695)
- `blackComponent` (page 696)
- `cyanComponent` (page 702)
- `magentaComponent` (page 708)
- `yellowComponent` (page 713)

Declared In

`NSColor.h`

getHue:saturation:brightness:alpha:

Returns the receiver's HSB component and opacity values in the respective arguments.

```
- (void)getHue:(CGFloat *)hue saturation:(CGFloat *)saturation brightness:(CGFloat
    *)brightness alpha:(CGFloat *)alpha
```

Parameters

hue

Upon return, contains the hue component of the color object.

saturation

Upon return, contains the saturation component of the color object.

brightness

Upon return, contains the brightness component of the color object.

alpha

Upon return, contains the opacity value of the color object.

Discussion

If `NULL` is passed in as an argument, the method doesn't set that value. This method works only with objects representing colors in the `NSCalibratedRGBColorSpace` or `NSDeviceRGBColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alphaComponent` (page 695)
- `brightnessComponent` (page 697)
- `hueComponent` (page 707)
- `saturationComponent` (page 710)

Related Sample Code
MenuItemView

Declared In
`NSColor.h`

getRed:green:blue:alpha:

Returns the receiver's RGB component and opacity values in the respective arguments.

```
- (void)getRed:(CGFloat *)red green:(CGFloat *)green blue:(CGFloat *)blue
    alpha:(CGFloat *)alpha
```

Parameters

red
Upon return, contains the red component of the color object.

green
Upon return, contains the green component of the color object.

blue
Upon return, contains the blue component of the color object.

alpha
Upon return, contains the opacity value of the color object.

Discussion
If `NULL` is passed in as an argument, the method doesn't set that value. This method works only with objects representing colors in the `NSCalibratedRGBColorSpace` or `NSDeviceRGBColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alphaComponent` (page 695)
- `blueComponent` (page 697)
- `greenComponent` (page 706)
- `redComponent` (page 710)

Related Sample Code
CWCocoaComponent
FunHouse
Quartz 2D Shadings
RadiantColorPicker

Reminders

Declared In
NSColor.h

getWhite:alpha:

Returns the receiver's grayscale value and alpha values.

```
- (void)getWhite:(CGFloat *)white alpha:(CGFloat *)alpha
```

Parameters

white

Upon return, contains the grayscale value of the color object.

alpha

Upon return, contains the opacity value of the color object.

Discussion
If NULL is passed in as an argument, the method doesn't set that value. This method works only with objects representing colors in the NSCalibratedWhiteColorSpace, NSCalibratedBlackColorSpace, NSDeviceBlackColorSpace, or NSDeviceWhiteColorSpace color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- alphaComponent (page 695)
- whiteComponent (page 712)

Declared In
NSColor.h

greenComponent

Returns the receiver's green component.

```
- (CGFloat)greenComponent
```

Return Value
The color object's green component.

Discussion
This method works only with objects representing colors in the NSCalibratedRGBColorSpace or NSDeviceRGBColorSpace color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- getRed:green:blue:alpha: (page 705)

Related Sample Code
CIColorTracking
Color Sampler
QTCoreVideo301
Sketch+Accessibility

Declared In
`NSColor.h`

highlightWithLevel:

Returns an `NSColor` object that represents a blend between the receiver and the highlight color returned by `highlightColor` (page 684).

```
- (NSColor *)highlightWithLevel:(CGFloat)highlightLevel
```

Parameters

highlightLevel

The amount of the highlight color that is blended with the receiver's color. This should be a number from 0.0 through 1.0. A *highlightLevel* below 0.0 is interpreted as 0.0; a *highlightLevel* above 1.0 is interpreted as 1.0.

Return Value
The new `NSColor` object. Returns `nil` if the colors can't be converted.

Discussion
Invoke this method when you want to brighten the receiving `NSColor` for use in highlights.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- shadowWithLevel:` (page 712)

Declared In
`NSColor.h`

hueComponent

Returns the hue component of the HSB color equivalent to the receiver.

```
- (CGFloat)hueComponent
```

Return Value
The color object's hue component.

Discussion
This method works only with objects representing colors in the `NSCalibratedRGBColorSpace` or `NSDeviceRGBColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- getHue:saturation:brightness:alpha: (page 704)

Declared In
NSColor.h

localizedCatalogNameComponent

Returns the name of the catalog containing the receiver's name as a localized string.

```
- (NSString *)localizedCatalogNameComponent
```

Return Value
The name of catalog containing the color object's name as a localized string. This string may be displayed in user interface items like color pickers.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ colorWithCatalogName:colorName: (page 672)
- colorNameComponent (page 698)

Declared In
NSColor.h

localizedColorNameComponent

Returns the name of the receiver as a localized string.

```
- (NSString *)localizedColorNameComponent
```

Return Value
The name of color object as a localized string. This string may be displayed in user interface items like color pickers.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ colorWithCatalogName:colorName: (page 672)
- catalogNameComponent (page 698)
- colorNameComponent (page 698)
- localizedCatalogNameComponent (page 708)

Declared In
NSColor.h

magentaComponent

Returns the receiver's magenta component.

```
- (CGFloat)magentaComponent
```

Return Value
The color object's magenta component.

Discussion
This method works only with objects representing colors in the `NSDeviceCMYKColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- getCyan:magenta:yellow:black:alpha:` (page 703)

Declared In
`NSColor.h`

numberOfComponents

Returns the number of components in the receiver.

```
- (NSInteger)numberOfComponents
```

Return Value
The number of components in the color object. The floating-point components counted include alpha. This method raises an exception if the receiver doesn't have floating-point components.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- colorSpace` (page 699)
`- getComponents:` (page 703)

Related Sample Code
AnimatedTableView

Declared In
`NSColor.h`

patternImage

Returns the image that the receiver is using as a pattern.

```
- (NSImage *)patternImage
```

Return Value
The image used by the color object. If the receiver doesn't have an image, this method raises an exception.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSColor.h

redComponent

Returns the receiver's red component.

```
- (CGFloat)redComponent
```

Return Value
The color object's red component.

Discussion
This method works only with objects representing colors in the `NSCalibratedRGBColorSpace` or `NSDeviceRGBColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- getRed:green:blue:alpha:` (page 705)

Related Sample Code
CIColorTracking
Color Sampler
QTCoreVideo301
Sketch+Accessibility

Declared In
NSColor.h

saturationComponent

Returns the saturation component of the HSB color equivalent to the receiver.

```
- (CGFloat)saturationComponent
```

Return Value
The color object's saturation component.

Discussion
This method works only with objects representing colors in the `NSCalibratedRGBColorSpace` or `NSDeviceRGBColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- getHue:saturation:brightness:alpha:` (page 704)

Declared In
NSColor.h

set

Sets the color of subsequent drawing to the color that the receiver represents.

```
- (void)set
```

Discussion
This method should be implemented in subclasses.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DockTile
FilterDemo
MatrixMixerTest
Sketch-112
WhackedTV

Declared In
`NSColor.h`

setFill

Sets the fill color of subsequent drawing to the receiver's color.

```
- (void)setFill
```

Discussion
This method should be implemented in subclasses.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setStroke` (page 711)

Related Sample Code
AnimatedTableView
JSPong
SampleRaster

Declared In
`NSColor.h`

setStroke

Sets the stroke color of subsequent drawing to the receiver's color.

```
- (void)setStroke
```

Discussion
This method should be implemented in subclasses.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setFill` (page 711)

Related Sample Code
SampleRaster

Declared In
`NSColor.h`

shadowWithLevel:

Returns an `NSColor` object that represents a blend between the receiver and the shadow color returned by `shadowColor` (page 692).

```
- (NSColor *)shadowWithLevel:(CGFloat)shadowLevel
```

Parameters

shadowLevel
The amount of the shadow color used for the blend. This should be a number from 0.0 through 1.0. A *shadowLevel* below 0.0 is interpreted as 0.0; a *shadowLevel* above 1.0 is interpreted as 1.0.

Return Value
The new `NSColor` object. Returns `nil` if the colors can't be converted.

Discussion
Invoke this method when you want to darken the receiving NSColor for use in shadows.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `highlightWithLevel:` (page 707)

Declared In
`NSColor.h`

whiteComponent

Returns the receiver's white component.

```
- (CGFloat)whiteComponent
```

Return Value
The color object's white component.

Discussion
This method works only with objects representing colors in the `NSCalibratedWhiteColorSpace`, `NSCalibratedBlackColorSpace`, `NSDeviceBlackColorSpace`, or `NSDeviceWhiteColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `getWhite:alpha:` (page 706)

Declared In
`NSColor.h`

writeToPasteboard:

Writes the receiver's data to the specified pasteboard.

```
- (void)writeToPasteboard:(NSPasteboard *)pasteBoard
```

Parameters

pasteBoard
The pasteboard to which to write the receiver's color data. If this pasteboard doesn't support color data, the method does nothing.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ `colorFromPasteboard:` (page 669)

Declared In
`NSColor.h`

yellowComponent

Returns the receiver's yellow component.

```
- (CGFloat)yellowComponent
```

Return Value
The color object's yellow component.

Discussion
This method works only with objects representing colors in the `NSDeviceCMYKColorSpace` color space. Sending it to other objects raises an exception.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `getCyan:magenta:yellow:black:alpha:` (page 703)

Declared In
`NSColor.h`

Constants

For definitions of `NSColor` constants, as well as a discussion of their usage, see "About Color Spaces" in *Color Programming Topics*.

AppKit Versions for NSColor Bug Fixes

The version of the AppKit framework containing a specific bug fix.

```
#define NSAppKitVersionNumberWithPatternColorLeakFix 641.0
```

Constants

`NSAppKitVersionNumberWithPatternColorLeakFix`

The specific version of the AppKit framework that introduced the fix for correctly autoreleasing objects returned by the `colorWithPatternImage:` (page 677) method. Developers should not need to use this constant unless they are writing applications for Mac OS X v10.1 and earlier.

Available in Mac OS X v10.2 and later.

Declared in `NSColor.h`.

Notifications

NSSystemColorsDidChangeNotification

Sent when the system colors have been changed (such as through a system control panel interface).

This notification contains no notification object and no *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColor.h`

NSColorList Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSColorList.h
Companion guide	Color Programming Topics
Related sample code	AnimatedTableView

Overview

An `NSColorList` object is an ordered list of `NSColor` objects, identified by keys. Instances of `NSColorList`, or more simply color lists, are used to manage named lists of `NSColor` instances. The `NSColorPanel` list mode color picker uses instances of `NSColorList` to represent any lists of colors that come with the system, as well as any lists created by the user. An application can use `NSColorList` to manage document-specific color lists.

Adopted Protocols

NSCoding

- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Initializing an NSColorList Object

`- initWithName:` (page 718)
Initializes and returns the receiver, registering it under the given name if it isn't in use already.

- `initWithName:fromFile:` (page 719)
  Initializes and returns the receiver, registering it under the given name if it isn't in use already.

Getting Color Lists

+ `availableColorLists` (page 717)
  Returns an array of all color lists found in the standard color list directories.

+ `colorListNamed:` (page 717)
  Searches the array that's returned by `availableColorLists` (page 717) and returns the color list with the given name.

Getting Color List Properties

- `name` (page 720)
  Returns the name of the receiver.

- `isEditable` (page 720)
  Returns a Boolean value indicating whether the receiver can be modified.

Managing Colors By Key

- `allKeys` (page 718)
  Returns an array of the keys by which the `NSColor` objects are stored in the receiver.

- `colorWithKey:` (page 718)
  Returns the `NSColor` object associated with the given key.

- `insertColor:key:atIndex:` (page 719)
  Inserts the specified color at the specified location in the receiver.

- `removeColorWithKey:` (page 721)
  Removes the color associated with the specified key from the receiver.

- `setColor:forKey:` (page 721)
  Associates the specified `NSColor` object with the specified key.

Writing and Removing Color-List Files

- `removeFile` (page 721)
  Removes the file from which the list was created, if the file is in a standard search path and owned by the user.

- `writeToFile:` (page 722)
  Saves the receiver to a file at the specified path.

Class Methods

availableColorLists

Returns an array of all color lists found in the standard color list directories.

```
+ (NSArray *)availableColorLists
```

Return Value
An array of `NSColorList` objects representing all of the color lists found in the standard color list directories, including color catalogs (lists of colors identified only by name). Color lists created at runtime aren't included in this list unless they're saved into one of the standard color list directories.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ colorListNamed:` (page 717)

Declared In
`NSColorList.h`

colorListNamed:

Searches the array that's returned by `availableColorLists` (page 717) and returns the color list with the given name.

```
+ (NSColorList *)colorListNamed:(NSString *)name
```

Parameters

name

The name of the color list to retrieve. This name must not include the "`.clr`" suffix.

Return Value
The color list with the specified name or `nil` if no such color list exists.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- name` (page 720)

Related Sample Code
AnimatedTableView

Declared In
`NSColorList.h`

Instance Methods

allKeys

Returns an array of the keys by which the `NSColor` objects are stored in the receiver.

```
- (NSArray *)allKeys
```

Return Value
An array of `NSString` objects containing all the keys by which the `NSColor` objects are stored in the receiver.

The length of this array equals the number of colors, and its contents are arranged according to the ordering specified when the colors were inserted.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AnimatedTableView

Declared In
`NSColorList.h`

colorWithKey:

Returns the `NSColor` object associated with the given key.

```
- (NSColor *)colorWithKey:(NSString *)key
```

Parameters

key
The key for which to retrieve the color.

Return Value
The color associated with the given key or `nil` if there is none.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AnimatedTableView

Declared In
`NSColorList.h`

initWithName:

Initializes and returns the receiver, registering it under the given name if it isn't in use already.

```
- (id)initWithName:(NSString *)name
```

Parameters

name

The name under which to register the color list. Specify @"" if you don't want a name.

Return Value
The initialized color list.

Discussion
This method invokes `initWithName:fromFile:` (page 719) with a `fromFile:` argument of `nil`, indicating that the color list doesn't need to be initialized from a file. Note that this method does not add the color list to `availableColorLists` (page 717) until the color list is saved into the user's path with `writeToFile:` (page 722) with a value of `nil`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorList.h`

initWithName:fromFile:

Initializes and returns the receiver, registering it under the given name if it isn't in use already.

```
- (id)initWithName:(NSString *)name fromFile:(NSString *)path
```

Parameters

name

The name of the file for the color list (minus the ".clr" extension). Specify @"" if you don't want a name.

path

The full path to the file for the color list. A `nil` path indicates the color list should be initialized with no colors.

Discussion
Note that this method does not add the color list to `availableColorLists` (page 717) until the color list is saved into the user's path with `writeToFile:` (page 722) with a value of `nil`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorList.h`

insertColor:key:atIndex:

Inserts the specified color at the specified location in the receiver.

```
- (void)insertColor:(NSColor *)color key:(NSString *)key atIndex:(NSUInteger)location
```

Parameters

color

The color to add to the color list.

key
The key with which to associate the color.

location
The location in the color list at which to place the specified color. Locations are numbered starting with 0.

Discussion
If the list already contains a color with the same key at a different location, it's removed from the old location. This method posts `NSColorListDidChangeNotification` (page 722) to the default notification center. It raises `NSColorListNotEditableException` if the color list isn't editable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `colorWithKey:` (page 718)
- `removeColorWithKey:` (page 721)
- `setColor:forKey:` (page 721)

Declared In
`NSColorList.h`

isEditable

Returns a Boolean value indicating whether the receiver can be modified.

```
- (BOOL)isEditable
```

Return Value
`YES` if the color list can be modified; otherwise `NO`. This result depends on the source of the list: If it came from a write-protected file, this method returns `NO`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorList.h`

name

Returns the name of the receiver.

```
- (NSString *)name
```

Return Value
The name of the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorList.h`

removeColorWithKey:

Removes the color associated with the specified key from the receiver.

```
- (void)removeColorWithKey:(NSString *)key
```

Parameters

key

The key for which to remove the color.

Discussion

This method does nothing if the receiver doesn't contain the key. This method posts `NSColorListDidChangeNotification` (page 722) to the default notification center. It raises `NSColorListNotEditableException` if the receiver is not editable.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `insertColor:key:atIndex:` (page 719)
- `setColor:forKey:` (page 721)

Declared In

`NSColorList.h`

removeFile

Removes the file from which the list was created, if the file is in a standard search path and owned by the user.

```
- (void)removeFile
```

Discussion

The receiver is removed from the list of available color lists returned by `availableColorLists` (page 717). If there are no outstanding references to the color list, this method might deallocate the object as well.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSColorList.h`

setColor:forKey:

Associates the specified `NSColor` object with the specified key.

```
- (void)setColor:(NSColor *)color forKey:(NSString *)key
```

Parameters

color

The color to associate with the given key.

key

The key.

Discussion
If the list already contains *key*, this method sets the corresponding color to *color*; otherwise, it inserts *color* at the end of the list by invoking `insertColor:key:atIndex:` (page 719).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `colorWithKey:` (page 718)
- `insertColor:key:atIndex:` (page 719)
- `removeColorWithKey:` (page 721)

Declared In
`NSColorList.h`

writeToFile:

Saves the receiver to a file at the specified path.

```
- (BOOL)writeToFile:(NSString *)path
```

Parameters
path

The path at which to save the color list. If *path* is a directory, the receiver is saved in a file named *listname*`.clr` in that directory (where *listname* is the name with which the receiver was initialized).

If *path* includes a filename, this method saves the file under that name. If *path* is `nil`, the file is saved as *listname*`.clr` in the user's private colorlists directory.

Return Value
`YES` upon success and `NO` if the method fails to write the file.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `removeFile` (page 721)

Declared In
`NSColorList.h`

Notifications

NSColorListDidChangeNotification

Posted whenever a color list changes. The notification object is the `NSColorList` object that changed. This notification does not contain a *userInfo* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorList.h`

NSColorPanel Class Reference

Inherits from	NSPanel : NSWindow : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSWindow) NSAnimatablePropertyContainer (NSWindow) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSColorPanel.h
Companion guide	Color Programming Topics
Related sample code	CWCocoaComponent QTAudioContextInsert QTAudioExtractionPanel QTCoreVideo301 QTKitPlayer

Overview

The `NSColorPanel` class provides a standard user interface for selecting color in an application. It provides a number of standard color selection modes and, with the `NSColorPickingDefault` and `NSColorPickingCustom` protocols, allows an application to add its own color selection modes. It allows the user to save swatches containing frequently used colors.

Tasks

Obtaining the Shared Color-Panel Object

`+ sharedColorPanel` (page 729)
Returns the shared `NSColorPanel` instance, creating it if necessary.

`+ sharedColorPanelExists` (page 729)
Returns a Boolean value indicating whether the `NSColorPanel` has been created already.

Configuring the Color Panel

- `accessoryView` (page 729)
  Returns the accessory view.
- `isContinuous` (page 731)
  Returns a Boolean value indicating whether the receiver continuously sends the action message to the target.
- `mode` (page 732)
  Returns the color picker mode of the receiver.
- `setAccessoryView:` (page 732)
  Sets the accessory view displayed in the receiver.
- `setAction:` (page 733)
  Sets the color panel's action message.
- `setContinuous:` (page 734)
  Sets the receiver to send the action message to its target continuously as the user sets the color.
- `setMode:` (page 734)
  Sets the mode of the receiver the mode is one of the modes allowed by the color mask.
- `setShowsAlpha:` (page 735)
  Tells the receiver whether or not to show alpha values and an opacity slider.
- `setTarget:` (page 735)
  Sets the target of the receiver.
- `showsAlpha` (page 736)
  Returns a Boolean value indicating whether or not the receiver shows alpha values and an opacity slider.

Managing Color Lists

- `attachColorList:` (page 730)
  Adds the list of `NSColor` objects specified to all the color pickers in the receiver that display color lists by invoking `attachColorList:` (page 730) on all color pickers in the application.
- `detachColorList:` (page 731)
  Removes the list of colors from all the color pickers in the receiver that display color lists by invoking `detachColorList:` on all color pickers in the application.

Setting Color Picker Modes

+ `setPickerMask:` (page 728)
  Determines which color selection modes are available in an application's `NSColorPanel`.
+ `setPickerMode:` (page 728)
  Specifies the color panel's initial picker.

Setting Color

`+ dragColor:withEvent:fromView:` (page 727)
Drags a color into a destination view from the specified source view.

`- setColor:` (page 733)
Sets the color of the receiver.

Getting Color Information

`- alpha` (page 730)
Returns the receiver's current alpha value based on its opacity slider.

`- color` (page 731)
Returns the currently selected color in the receiver.

Responding to a Color Change

`- changeColor:` (page 736) *delegate method*
Sent to the first responder when the user selects a color in an `NSColorPanel` object.

Class Methods

dragColor:withEvent:fromView:

Drags a color into a destination view from the specified source view.

```
+ (BOOL)dragColor:(NSColor *)color withEvent:(NSEvent *)anEvent fromView:(NSView
    *)sourceView
```

Parameters

color
The color to drag.

anEvent
The drag event.

sourceView
The view from which the color was dragged.

Return Value
`YES`

Discussion
This method is usually invoked by the `mouseDown:` method of *sourceView*. The dragging mechanism handles all subsequent events.

Because it is a class method, `dragColor:withEvent:fromView:` can be invoked whether or not the instance of `NSColorPanel` exists.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorPanel.h`

setPickerMask:

Determines which color selection modes are available in an application's `NSColorPanel`.

```
+ (void)setPickerMask:(NSUInteger)mask
```

Parameters

mask

One or more logically ORed color mode masks described in Color Panel Mode Masks (page 737).

Discussion
This method has an effect only before an `NSColorPanel` object is instantiated.

If you create a class that implements the color-picking protocols (`NSColorPickingDefault` and `NSColorPickingCustom`), you may want to give it a unique mask—one different from those defined for the standard color pickers. To display your color picker, your application will need to logically OR that unique mask with the standard color mask constants when invoking this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ setPickerMode:` (page 728)

Declared In
`NSColorPanel.h`

setPickerMode:

Specifies the color panel's initial picker.

```
+ (void)setPickerMode:(NSColorPanelMode)mode
```

Parameters

mode

A constant specifying which color picker mode is initially visible. This is one of the symbolic constants described in Color Panel Modes (page 738).

Discussion
This method may be called at any time, whether or not an application's `NSColorPanel` has been instantiated.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ setPickerMask:` (page 728)
`- setMode:` (page 734)

Declared In
NSColorPanel.h

sharedColorPanel

Returns the shared NSColorPanel instance, creating it if necessary.

```
+ (NSColorPanel *)sharedColorPanel
```

Return Value
The shared NSColorPanel instance.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AnimatingViews
CWCocoaComponent
FunHouse
QTAudioExtractionPanel
QTCoreVideo301

Declared In
NSColorPanel.h

sharedColorPanelExists

Returns a Boolean value indicating whether the NSColorPanel has been created already.

```
+ (BOOL)sharedColorPanelExists
```

Return Value
YES if the NSColorPanel has been created already; otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ sharedColorPanel (page 729)

Declared In
NSColorPanel.h

Instance Methods

accessoryView

Returns the accessory view.

```
- (NSView *)accessoryView
```

Return Value
The accessory view or `nil` if there is none.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAccessoryView:` (page 732)

Declared In
`NSColorPanel.h`

alpha

Returns the receiver's current alpha value based on its opacity slider.

```
- (CGFloat)alpha
```

Return Value
The alpha value of the `NSColorPanel`. This is 1.0 (opaque) if the panel has no opacity slider.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setShowsAlpha:` (page 735)
- `showsAlpha` (page 736)

Declared In
`NSColorPanel.h`

attachColorList:

Adds the list of `NSColor` objects specified to all the color pickers in the receiver that display color lists by invoking `attachColorList:` (page 730) on all color pickers in the application.

```
- (void)attachColorList:(NSColorList *)colorList
```

Parameters
colorList
The list of colors to add to the color pickers in the receiver.

Discussion
An application should use this method to add an `NSColorList` saved with a document in its file package or in a directory other than `NSColorList`'s standard search directories.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `detachColorList:` (page 731)

Declared In
NSColorPanel.h

color

Returns the currently selected color in the receiver.

```
- (NSColor *)color
```

Return Value
The currently selected color.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setColor: (page 733)

Declared In
NSColorPanel.h

detachColorList:

Removes the list of colors from all the color pickers in the receiver that display color lists by invoking detachColorList: on all color pickers in the application.

```
- (void)detachColorList:(NSColorList *)colorList
```

Parameters
colorList
The list of NSColor objects to remove from the color pickers in the color panel.

Discussion
Your application should use this method to remove an NSColorList saved with a document in its file package or in a directory other than NSColorList's standard search directories.

Availability
Available in Mac OS X v10.0 and later.

See Also
- attachColorList: (page 730)

Declared In
NSColorPanel.h

isContinuous

Returns a Boolean value indicating whether the receiver continuously sends the action message to the target.

```
- (BOOL)isContinuous
```

Return Value
`YES` if the receiver continuously sends the action message to the target as the user manipulates the color picker; otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setContinuous:` (page 734)

Declared In
`NSColorPanel.h`

mode

Returns the color picker mode of the receiver.

```
- (NSColorPanelMode)mode
```

Return Value
A constant indicating the current color picker mode. See Color Panel Modes (page 738).

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ setPickerMode:` (page 728)
`- setMode:` (page 734)

Declared In
`NSColorPanel.h`

setAccessoryView:

Sets the accessory view displayed in the receiver.

```
- (void)setAccessoryView:(NSView *)aView
```

Parameters
aView
The accessory view displayed in the receiver. The accessory view can be any custom view you want to display with `NSColorPanel`, such as a view offering color blends in a drawing program. The accessory view is displayed below the color picker and above the color swatches in the `NSColorPanel`. The `NSColorPanel` automatically resizes to accommodate the accessory view.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- accessoryView` (page 729)

Related Sample Code
QTCoreVideo301

Declared In
NSColorPanel.h

setAction:

Sets the color panel's action message.

```
- (void)setAction:(SEL)action
```

Return Value
The action message. When you select a color in the color panel `NSColorPanel` sends its action to its target, provided that neither the action nor the target is `nil`. The action is `NULL` by default.

Discussion
See *Action Messages* for additional information on action messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTarget:` (page 735)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer

Declared In
NSColorPanel.h

setColor:

Sets the color of the receiver.

```
- (void)setColor:(NSColor *)color
```

Parameters
color
The color of the `NSColorPanel`.

Discussion
This method posts an `NSColorPanelColorDidChangeNotification` (page 740) with the receiver to the default notification center.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- color` (page 731)

Related Sample Code
QTAudioContextInsert

QTAudioExtractionPanel
QTKitImport
QTKitPlayer

Declared In
NSColorPanel.h

setContinuous:

Sets the receiver to send the action message to its target continuously as the user sets the color.

```
- (void)setContinuous:(BOOL)flag
```

Parameters

flag

YES to have the receiver send its action message to its target continuously as the color of the NSColorPanel is set by the user; otherwise NO. Set this to YES if, for example, you want to continuously update the color of the target.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isContinuous (page 731)

Declared In
NSColorPanel.h

setMode:

Sets the mode of the receiver the mode is one of the modes allowed by the color mask.

```
- (void)setMode:(NSColorPanelMode)mode
```

Parameters

mode

A constant specifying the mode of the color panel. These constants are described in Color Picker Modes (page 738). The color mask is set when you first create the shared instance of NSColorPanel for an application.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ setPickerMode: (page 728)
- mode (page 732)

Declared In
NSColorPanel.h

setShowsAlpha:

Tells the receiver whether or not to show alpha values and an opacity slider.

```
- (void)setShowsAlpha:(BOOL)flag
```

Parameters

flag

YES to have the color panel show alpha values and an opacity slider; otherwise NO.

Discussion

Note that calling the NSColor method setIgnoresAlpha: (page 691) with a value of YES overrides any value set with this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- alpha (page 730)
- showsAlpha (page 736)

Related Sample Code

AnimatingViews
CWCocoaComponent
FunHouse
Quartz 2D Shadings

Declared In

NSColorPanel.h

setTarget:

Sets the target of the receiver.

```
- (void)setTarget:(id)target
```

Parameters

target

The target of the receiver. When you select a color in the color panel NSColorPanel sends its action to its target, provided that neither the action nor the target is nil. The target is nil by default.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setAction: (page 733)
- setContinuous: (page 734)

Related Sample Code

QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer

Declared In
NSColorPanel.h

showsAlpha

Returns a Boolean value indicating whether or not the receiver shows alpha values and an opacity slider.

```
- (BOOL)showsAlpha
```

Return Value
YES if the color picker shows alpha values and an opacity slider; otherwise NO.

Discussion
Note that calling the NSColor method setIgnoresAlpha: (page 691) with a value of YES overrides any value set with setShowsAlpha: (page 735).

Availability
Available in Mac OS X v10.0 and later.

See Also
- alpha (page 730)
- setShowsAlpha: (page 735)

Declared In
NSColorPanel.h

Delegate Methods

changeColor:

Sent to the first responder when the user selects a color in an NSColorPanel object.

```
- (void)changeColor:(id)sender
```

Parameters
sender
The NSColorPanel sending the message.

Discussion
When the user selects a color in an NSColorPanel object, the panel sends a changeColor: action message to the first responder. You can override this method in any responder that needs to respond to a color change.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSColorPanel.h

Constants

Color Picker Mode Masks

Used to specify in the `setPickerMask:` (page 728) class method which of the color modes the `NSColorPanel` can use.

```
enum {
    NSColorPanelGrayModeMask            = 0x00000001,
    NSColorPanelRGBModeMask             = 0x00000002,
    NSColorPanelCMYKModeMask            = 0x00000004,
    NSColorPanelHSBModeMask             = 0x00000008,
    NSColorPanelCustomPaletteModeMask   = 0x00000010,
    NSColorPanelColorListModeMask       = 0x00000020,
    NSColorPanelWheelModeMask           = 0x00000040,
    NSColorPanelCrayonModeMask          = 0x00000080,
    NSColorPanelAllModesMask            = 0x0000ffff
};
```

Constants

`NSColorPanelGrayModeMask`

Grayscale-alpha.

Available in Mac OS X v10.0 and later.

Declared in `NSColorPanel.h`.

`NSColorPanelRGBModeMask`

Red-green-blue.

Available in Mac OS X v10.0 and later.

Declared in `NSColorPanel.h`.

`NSColorPanelCMYKModeMask`

Cyan-yellow-magenta-black.

Available in Mac OS X v10.0 and later.

Declared in `NSColorPanel.h`.

`NSColorPanelHSBModeMask`

Hue-saturation-brightness.

Available in Mac OS X v10.0 and later.

Declared in `NSColorPanel.h`.

`NSColorPanelCustomPaletteModeMask`

Custom palette.

Available in Mac OS X v10.0 and later.

Declared in `NSColorPanel.h`.

`NSColorPanelColorListModeMask`

Custom color list.

Available in Mac OS X v10.0 and later.

Declared in `NSColorPanel.h`.

`NSColorPanelWheelModeMask`
: Color wheel.

  Available in Mac OS X v10.0 and later.

  Declared in `NSColorPanel.h`.

`NSColorPanelCrayonModeMask`
: Crayons.

  Declared in `NSColorPanel.h`.

  Available in Mac OS X v10.2 and later.

`NSColorPanelAllModesMask`
: All of the above.

  Available in Mac OS X v10.0 and later.

  Declared in `NSColorPanel.h`.

Discussion
For more information, see "Choosing the Color Pickers in a Color Panel".

Declared In
`NSColorPanel.h`

NSColorPanelMode

A type defined for the `enum` constants specifying color panel modes.

```
typedef NSInteger NSColorPanelMode;
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSColorPanel.h`

Color Panel Modes

Specify the active color mode used when an application's instance of `NSColorPanel` is masked for more than one color mode.

```
enum {
    NSNoModeColorPanel              = -1,
    NSGrayModeColorPanel             = 0,
    NSRGBModeColorPanel              = 1,
    NSCMYKModeColorPanel             = 2,
    NSHSBModeColorPanel              = 3,
    NSCustomPaletteModeColorPanel    = 4,
    NSColorListModeColorPanel        = 5,
    NSWheelModeColorPanel            = 6,
    NSCrayonModeColorPanel           = 7
};
```

Constants

`NSNoModeColorPanel`
Indicates no color panel mode.
Available in Mac OS X version 10.5 and later.
Declared in `NSColorPanel.h`.

`NSGrayModeColorPanel`
Grayscale-alpha
Available in Mac OS X v10.0 and later.
Declared in `NSColorPanel.h`.

`NSRGBModeColorPanel`
Red-green-blue
Available in Mac OS X v10.0 and later.
Declared in `NSColorPanel.h`.

`NSCMYKModeColorPanel`
Cyan-yellow-magenta-black
Available in Mac OS X v10.0 and later.
Declared in `NSColorPanel.h`.

`NSHSBModeColorPanel`
Hue-saturation-brightness
Available in Mac OS X v10.0 and later.
Declared in `NSColorPanel.h`.

`NSCustomPaletteModeColorPanel`
Custom palette
Available in Mac OS X v10.0 and later.
Declared in `NSColorPanel.h`.

`NSColorListModeColorPanel`
Custom color list
Available in Mac OS X v10.0 and later.
Declared in `NSColorPanel.h`.

`NSWheelModeColorPanel`
Color wheel
Available in Mac OS X v10.0 and later.
Declared in `NSColorPanel.h`.

`NSCrayonModeColorPanel`
: Crayons.

  Declared in `NSColorPanel.h`.

  Available in Mac OS X v10.2 and later.

Discussion
These enum constants are specified or returned in the instance methods `mode` (page 732) and `setMode:` (page 734), and in the `setPickerMode:` (page 728)class method. For more information, see "Choosing the Color Pickers in a Color Panel".

Declared In
`NSColorPanel.h`

Notifications

NSColorPanelColorDidChangeNotification

Posted when the color of the `NSColorPanel` is set, as when `setColor:` (page 733) is invoked.

The notification object is the notifying `NSColorPanel`. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorPanel.h`

NSColorPicker Class Reference

Inherits from	NSObject
Conforms to	NSColorPickingDefault NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSColorPicker.h
Companion guide	Color Programming Topics
Related sample code	MyCustomColorPicker RadiantColorPicker

Overview

The `NSColorPicker` class is an abstract superclass that implements the `NSColorPickingDefault` protocol. The `NSColorPickingDefault` and `NSColorPickingCustom` protocols define a way to add color pickers (custom user interfaces for color selection) to the `NSColorPanel`.

Adopted Protocols

NSColorPickingDefault

- `- alphaControlAddedOrRemoved:` (page 3618)
- `- attachColorList:` (page 3618)
- `- detachColorList:` (page 744)
- `- initWithPickerMask:colorPanel:` (page 3620)
- `- insertNewButtonImage:in:` (page 3621)
- `- provideNewButtonImage` (page 3621)
- `- setMode:` (page 3622)
- `- viewSizeChanged:` (page 746)

Tasks

Initializing an NSColorPicker Object

- `initWithPickerMask:colorPanel:` (page 744)
  Initializes the color picker with the specified color panel and color picker mode mask.

Getting the Color Panel

- `colorPanel` (page 743)
  Returns the NSColorPanel that owns the receiver.

Adding Button Images

- `insertNewButtonImage:in:` (page 745)
  Sets the image used for the specified button cell.
- `provideNewButtonImage` (page 746)
  Returns the button image for the receiver.

Setting the Mode

- `setMode:` (page 746)
  Does nothing. Override to set the color picker's mode.

Mananging Color Lists

- `attachColorList:` (page 743)
  Does nothing. Override to attach a color list to a color picker.
- `detachColorList:` (page 744)
  Does nothing. Override to detach a color list from a color picker.

Responding to View Changes

- `viewSizeChanged:` (page 746)
  Does nothing. Override to respond to a size change.

Customizing the Color Picker

- `buttonToolTip` (page 743)
  Returns the tool tip to be shown when the mouse cursor is over the receiver's button image.

- minContentSize (page 745)
  Returns the minimum content size for the receiver.

Instance Methods

attachColorList:

Does nothing. Override to attach a color list to a color picker.

```
- (void)attachColorList:(NSColorList *)colorList
```

Parameters

colorList
The color list to attach to the color picker.

Availability
Available in Mac OS X v10.0 and later.

See Also
- detachColorList: (page 744)

Declared In
NSColorPicker.h

buttonToolTip

Returns the tool tip to be shown when the mouse cursor is over the receiver's button image.

```
- (NSString *)buttonToolTip
```

Return Value
A string representing the tool tip.

Discussion
Override this method to provide a custom tool tip. The default implementation of this method returns the name of the receiver's class. If you want the color picker to have no tool tip, return an empty string.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
RadiantColorPicker

Declared In
NSColorPicker.h

colorPanel

Returns the NSColorPanel that owns the receiver.

```
- (NSColorPanel *)colorPanel
```

Return Value
The owning color panel.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSColorPicker.h

detachColorList:

Does nothing. Override to detach a color list from a color picker.

```
- (void)detachColorList:(NSColorList *)colorList
```

Parameters

colorList
The color list to detach.

Availability
Available in Mac OS X v10.0 and later.

See Also
- attachColorList: (page 743)

Declared In
NSColorPicker.h

initWithPickerMask:colorPanel:

Initializes the color picker with the specified color panel and color picker mode mask.

```
- (id)initWithPickerMask:(NSUInteger)mask colorPanel:(NSColorPanel *)owningColorPanel
```

Parameters

mask
The color picker mask.

owningColorPanel
The NSColorPanel that owns the color picker. This value is cached so it can be returned later by the colorPanel (page 743) method.

Return Value
An initialized color picker object.

Discussion
Override this method to respond to the values in *mask* or do other custom initialization. If you override this method in a subclass, you should forward the message to super as part of the implementation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- colorPanel (page 743)

Declared In
NSColorPicker.h

insertNewButtonImage:in:

Sets the image used for the specified button cell.

```
- (void)insertNewButtonImage:(NSImage *)newButtonImage in:(NSButtonCell *)buttonCell
```

Parameters

newButtonImage
The image used for the specified button cell.

buttonCell
The button cell for which to set the image.

Discussion
Called by the color panel to insert a new image into the specified cell by invoking NSButtonCell's setImage: (page 69) method. Override this method to customize *newButtonImage* before insertion in *buttonCell*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- provideNewButtonImage (page 746)

Declared In
NSColorPicker.h

minContentSize

Returns the minimum content size for the receiver.

```
- (NSSize)minContentSize
```

Return Value
The minimum size of the receiver, an NSColorPicker object. The NSColorPanel object does not allow the color picker to be made smaller than this size.

Discussion
Override this method to return a minimum size for the color picker's content area. The default implementation of this method obtains the minimum content size from the view-autoresizing behavior specified for the receiver and returns that. You should not have to override this method if you properly set up the color picker's auto-sizing attributes in Interface Builder.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSColorPicker.h

provideNewButtonImage

Returns the button image for the receiver.

```
- (NSImage *)provideNewButtonImage
```

Return Value
The image placed on the mode button the user uses to select this color picker. This is the same image the color panel uses as an argument when sending the `insertNewButtonImage:in:` (page 745) message.) The default implementation looks in the color picker's bundle for a TIFF file named after the color picker's class, with the extension "`.tiff`".

Availability
Available in Mac OS X v10.0 and later.

See Also
`- insertNewButtonImage:in:` (page 745)

Related Sample Code
MyCustomColorPicker

Declared In
`NSColorPicker.h`

setMode:

Does nothing. Override to set the color picker's mode.

```
- (void)setMode:(NSColorPanelMode)mode
```

Parameters

mode

A constant specifying the color picking mode. These constants are defined in `AppKit/NSColorPanel.h`.

Discussion
In grayscale-alpha, red-green-blue, cyan-magenta-yellow-black, and hue-saturation-brightness modes, the user adjusts colors by manipulating sliders. In the custom palette mode, the user can load an `NSImage` file (TIFF or EPS) into the `NSColorPanel`, then select colors from the image. In custom color list mode, the user can create and load lists of named colors. The two custom modes provide `NSPopUpLists` for loading and saving files. Finally, color wheel mode provides a simplified control for selecting colors.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorPicker.h`

viewSizeChanged:

Does nothing. Override to respond to a size change.

```
- (void)viewSizeChanged:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorPicker.h`

NSColorSpace Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	AppKit/AppKit.h
Companion guide	Color Programming Topics
Related sample code	AnimatedTableView Quartz 2D Shadings Quartz Composer QCTV

Overview

The `NSColorSpace` class enables the creation of objects representing custom color spaces. You can make custom color spaces from ColorSync profiles or from ICC profiles. `NSColorSpace` also has factory methods that return objects representing the system color spaces.

You can use the `colorWithColorSpace:components:count:` (page 673) method of the `NSColor` class to create color objects using custom `NSColorSpace` objects. You can also send the `colorUsingColorSpace:` (page 700) message to an `NSColor` object to convert it between two color spaces, either of which may be a custom color space.

Tasks

Getting a Named NSColorSpace Object

`+ deviceRGBColorSpace` (page 752)
Returns an `NSColorSpace` object representing a calibrated or device-dependent RGB color space.

`+ genericRGBColorSpace` (page 754)
Returns an `NSColorSpace` object representing a device-independent RGB color space.

`+ deviceCMYKColorSpace` (page 751)
Returns an `NSColorSpace` object representing a calibrated or device-dependent CMYK color space.

+ genericCMYKColorSpace (page 753)
Returns an NSColorSpace object representing a device-independent CMYK color space.

+ deviceGrayColorSpace (page 752)
Returns an NSColorSpace object representing a calibrated or device-dependent gray color space.

+ genericGrayColorSpace (page 753)
Returns an NSColorSpace object representing a device-independent gray color space.

+ sRGBColorSpace (page 754)
Returns an NSColorSpace object representing an sRGB color space.

+ genericGamma22GrayColorSpace (page 753)
Returns an NSColorSpace object representing a gray color space with a gamma value of 2.2.

+ adobeRGB1998ColorSpace (page 751)
Returns an NSColorSpace object representing an Adobe RGB (1998) color space.

Getting the Color Spaces Available On the System

+ availableColorSpacesWithModel: (page 751)
Returns the list of color spaces available on the system that are displayed in the color panel, in the order they are displayed in the color panel.

Initializing a Custom NSColorSpace Object

- initWithCGColorSpace: (page 756)
Initializes and returns an NSColorSpace object initialized from a Core Graphics color-space object.

- initWithColorSyncProfile: (page 757)
Initializes and returns an NSColorSpace object given a ColorSync profile.

- initWithICCProfileData: (page 757)
Initializes and returns an NSColorSpace object given an ICC profile.

Accessing Color-Space Data and Attributes

- CGColorSpace (page 755)
Returns a Core Graphics color-space object that represents a color space equivalent to the receiver's.

- colorSpaceModel (page 755)
Returns the model on which the color space of the receiver is based.

- colorSyncProfile (page 755)
Returns the ColorSync profile from which the receiver was created.

- ICCProfileData (page 756)
Returns the ICC profile data from which the receiver was created.

- localizedName (page 758)
Returns the localized name of the receiver.

- numberOfColorComponents (page 758)
Returns the number of components supported by the receiver.

Class Methods

adobeRGB1998ColorSpace

Returns an `NSColorSpace` object representing an Adobe RGB (1998) color space.

```
+ (NSColorSpace *)adobeRGB1998ColorSpace
```

Return Value
The `NSColorSpace` object. This color-additive color space has red, green, blue, and alpha components.

Discussion
The Adobe RGB (1998) color space was designed to encompass most of the colors achievable on CMYK color printers, but by using RGB primary colors on a device such as the computer display. For more information on this color space, go to http://www.adobe.com/digitalimag/adobergb.html.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSColorSpace.h`

availableColorSpacesWithModel:

Returns the list of color spaces available on the system that are displayed in the color panel, in the order they are displayed in the color panel.

```
+ (NSArray *)availableColorSpacesWithModel:(NSColorSpaceModel)model
```

Parameters

model

The model to return the color spaces for.

Return Value
The list of color spaces, or an empty array if no color spaces are available for the specified model.

Discussion
This method doesn't return color spaces created on the fly or spaces without user-displayable names. Pass `NSUnknownColorSpaceModel` as *model* to get all available color spaces.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSColorSpace.h`

deviceCMYKColorSpace

Returns an `NSColorSpace` object representing a calibrated or device-dependent CMYK color space.

```
+ (NSColorSpace *)deviceCMYKColorSpace
```

Return Value
The `NSColorSpace` object. This color space has cyan, magenta, yellow, black, and alpha components. Typical devices that use the color-subtractive CMYK color space are color printers. This object corresponds to the Cocoa color space name `NSDeviceCMYKColorSpace`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ genericCMYKColorSpace` (page 753)

Declared In
`NSColorSpace.h`

deviceGrayColorSpace

Returns an `NSColorSpace` object representing a calibrated or device-dependent gray color space.

```
+ (NSColorSpace *)deviceGrayColorSpace
```

Return Value
The `NSColorSpace` object. The color space also includes an alpha component. Typical devices that use this color space are grayscale printers and displays. This object corresponds to the Cocoa color space name `NSDeviceWhiteColorSpace`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ genericGrayColorSpace` (page 753)
`+ genericGamma22GrayColorSpace` (page 753)

Declared In
`NSColorSpace.h`

deviceRGBColorSpace

Returns an `NSColorSpace` object representing a calibrated or device-dependent RGB color space.

```
+ (NSColorSpace *)deviceRGBColorSpace
```

Return Value
The `NSColorSpace` object. This color space has red, green, blue, and alpha components. Typical devices that use the color-additive RGB color space are displays and scanners. This object corresponds to the Cocoa color space name `NSDeviceRGBColorSpace`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ genericRGBColorSpace` (page 754)

Declared In
NSColorSpace.h

genericCMYKColorSpace

Returns an NSColorSpace object representing a device-independent CMYK color space.

```
+ (NSColorSpace *)genericCMYKColorSpace
```

Return Value
The NSColorSpace object. This color space has cyan, magenta, yellow, black and alpha component.

Availability
Available in Mac OS X v10.4 and later.

See Also
+ deviceCMYKColorSpace (page 751)

Declared In
NSColorSpace.h

genericGamma22GrayColorSpace

Returns an NSColorSpace object representing a gray color space with a gamma value of 2.2.

```
+ (NSColorSpace *)genericGamma22GrayColorSpace
```

Return Value
The NSColorSpace object.

Availability
Available in Mac OS X v10.6 and later.

See Also
+ deviceGrayColorSpace (page 752)
+ genericGrayColorSpace (page 753)

Declared In
NSColorSpace.h

genericGrayColorSpace

Returns an NSColorSpace object representing a device-independent gray color space.

```
+ (NSColorSpace *)genericGrayColorSpace
```

Return Value
The NSColorSpace object. The color space also includes an alpha component. This object corresponds to the Cocoa color space name NSCalibratedWhiteColorSpace.

Availability
Available in Mac OS X v10.4 and later.

See Also
+ deviceGrayColorSpace (page 752)
+ genericGamma22GrayColorSpace (page 753)

Declared In
NSColorSpace.h

genericRGBColorSpace

Returns an NSColorSpace object representing a device-independent RGB color space.

```
+ (NSColorSpace *)genericRGBColorSpace
```

Return Value
The NSColorSpace object. This color-additive color space has red, green, blue, and alpha components. This object corresponds to the Cocoa color space name NSCalibratedRGBColorSpace.

Availability
Available in Mac OS X v10.4 and later.

See Also
+ deviceRGBColorSpace (page 752)

Related Sample Code
Quartz 2D Shadings
Quartz Composer QCTV

Declared In
NSColorSpace.h

sRGBColorSpace

Returns an NSColorSpace object representing an sRGB color space.

```
+ (NSColorSpace *)sRGBColorSpace
```

Return Value
The NSColorSpace object. This color-additive color space has red, green, blue, and alpha components.

Discussion
The sRGB color space is a standard color space for use on monitors, printers, and the Internet. For further information on sRGB, see http://www.color.org/srgb.html.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSColorSpace.h

Instance Methods

CGColorSpace

Returns a Core Graphics color-space object that represents a color space equivalent to the receiver's.

```
- (CGColorSpaceRef)CGColorSpace
```

Return Value
A reference to an Core Graphics color-space object (`CGColorSpaceRef`) or `NULL` if the type of color space represented by the receiver cannot be represented by a `CGColorSpace` object.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- initWithCGColorSpace:` (page 756)

Declared In
`NSColorSpace.h`

colorSpaceModel

Returns the model on which the color space of the receiver is based.

```
- (NSColorSpaceModel)colorSpaceModel
```

Return Value
A constant specifying the color space model of the receiver. See Color Space Models (page 758) for a list of valid `NSColorSpaceModel` constants.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSColorSpace.h`

colorSyncProfile

Returns the ColorSync profile from which the receiver was created.

```
- (void *)colorSyncProfile
```

Return Value
The ColorSync profile on which the receiver is based. You need to cast this value to an object of opaque type `CMProfileRef`. Returns `NULL` if the receiver was created from a ICC-profile data instead. See *ColorSync Manager Reference* for further information on `CMProfileRef`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `initWithColorSyncProfile:` (page 757)

Declared In
`NSColorSpace.h`

ICCProfileData

Returns the ICC profile data from which the receiver was created.

```
- (NSData *)ICCProfileData
```

Return Value
The ICC profile from which the receiver was created. This method attempts to compute the profile data from a `CMProfileRef` object and returns `nil` if it is unable to.

For information on ICC profiles, see the latest ICC specification at the International Color Consortium website.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `initWithICCProfileData:` (page 757)

Declared In
`NSColorSpace.h`

initWithCGColorSpace:

Initializes and returns an `NSColorSpace` object initialized from a Core Graphics color-space object.

```
- (id)initWithCGColorSpace:(CGColorSpaceRef)cgColorSpace
```

Parameters
cgColorSpace
A reference to a Core Graphics color-space object (`CGColorSpaceRef`).

Return Value
The initialized `NSColorSpace` object or `nil` if initialization was not successful, which might happen if the color space represented by the `CGColorSpace` object is not supported by `NSColorSpace`.

Discussion
Because `NSColorSpace` might retain or copy the `CGColorSpace` object depending on circumstances, you should not assume pointer equality of the provided object with that returned by `CGColorSpace` (page 755). And even if the pointer equality is preserved during runtime, it may not be after the `NSColorSpace` object is archived and unarchived.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
AnimatedTableView

Declared In
NSColorSpace.h

initWithColorSyncProfile:

Initializes and returns an NSColorSpace object given a ColorSync profile.

```
- (id)initWithColorSyncProfile:(void *)prof
```

Parameters

prof

The ColorSync profile to use when initializing the NSColorSpace object. This should be an object of opaque type CMProfileRef. See *ColorSync Manager Reference* for further information on CMProfileRef.

Return Value
The initialized NSColorSpace object or nil if initialization was not successful.

Availability
Available in Mac OS X v10.4 and later.

See Also
- colorSyncProfile (page 755)

Declared In
NSColorSpace.h

initWithICCProfileData:

Initializes and returns an NSColorSpace object given an ICC profile.

```
- (id)initWithICCProfileData:(NSData *)iccData
```

Parameters

iccData

The ICC profile to use when initializing the NSColorSpace object. For information on ICC profiles, see the latest ICC specification at the International Color Consortium website.

Return Value
The initialized NSColorSpace object or nil if initialization was not successful.

Availability
Available in Mac OS X v10.4 and later.

See Also
- ICCProfileData (page 756)

Declared In
NSColorSpace.h

localizedName

Returns the localized name of the receiver.

```
- (NSString *)localizedName
```

Return Value
The name of the color space as a localized string or `nil` if no localized name exists.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSColorSpace.h`

numberOfColorComponents

Returns the number of components supported by the receiver.

```
- (NSInteger)numberOfColorComponents
```

Return Value
The number of components (excluding alpha) the receiver supports or zero if the receiver is not based on `float` components.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSColorSpace.h`

Constants

NSColorSpaceModel

The type of the color-space mode constants listed in "Color Space Models" (page 758).

```
typedef NSInteger NSColorSpaceModel;
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSColorSpace.h`

Color Space Models

Identify the abstract model on which an `NSColorSpace` object is based.

```
typedef enum {
    NSUnknownColorSpaceModel = -1,
    NSGrayColorSpaceModel,
    NSRGBColorSpaceModel,
    NSCMYKColorSpaceModel,
    NSLABColorSpaceModel,
    NSDeviceNColorSpaceModel,
    NSIndexedColorSpaceModel,
    NSPatternColorSpaceModel
};
```

Constants

`NSUnknownColorSpaceModel`
This model is not known to `NSColorSpace`.

Available in Mac OS X v10.4 and later.

Declared in `NSColorSpace.h`.

`NSGrayColorSpaceModel`
The grayscale color-space model. Can refer to both device-dependent and generic color space variants.

Available in Mac OS X v10.4 and later.

Declared in `NSColorSpace.h`.

`NSRGBColorSpaceModel`
The RGB (red green blue) color-space model. Can refer to both device-dependent and generic color space variants.

Available in Mac OS X v10.4 and later.

Declared in `NSColorSpace.h`.

`NSCMYKColorSpaceModel`
The CYMK (cyan, yellow, magenta, black) color-space model. Can refer to both device-dependent and generic color space variants.

Available in Mac OS X v10.4 and later.

Declared in `NSColorSpace.h`.

`NSLABColorSpaceModel`
The L*a*b* device-independent color-space model, which represents colors relative to a reference white point.

Available in Mac OS X v10.4 and later.

Declared in `NSColorSpace.h`.

`NSDeviceNColorSpaceModel`
DeviceN is a color-space model from Adobe Systems, Inc. used in PostScript and PDF color specification.

Available in Mac OS X v10.4 and later.

Declared in `NSColorSpace.h`.

`NSIndexedColorSpaceModel`
An indexed color space, which identifies specified discrete colors in a color list by index number. An indexed color value (a color specification in indexed color space) consists of an index value that refers to a color in a color list.

Available in Mac OS X v10.5 and later.

Declared in `NSColorSpace.h`.

`NSPatternColorSpaceModel`

Identifies a pattern color space, which is simply an image that is repeated over and over again in a tiled pattern.

Available in Mac OS X v10.5 and later.

Declared in `NSColorSpace.h`.

Discussion

These constants are returned from `colorSpaceModel` (page 755) and are derived from the profile data encapsulated by the object.

Declared In

`NSColorSpace.h`

NSColorWell Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSColorWell.h
Companion guide	Color Programming Topics
Related sample code	ButtonMadness PDF Annotation Editor Quartz 2D Shadings Quartz Composer QCTV Sketch-112

Overview

An `NSColorWell` object is an `NSControl` for selecting and displaying a single color value. An example of an `NSColorWell` object (or simply color well) is found in an `NSColorPanel`, which uses a color well to display the current color selection. A color well is available from the Palettes panel of Interface Builder.

Tasks

Managing Color From Color Wells

`- color` (page 763)
Returns the color of the receiver.

`- setColor:` (page 765)
Sets the color of the receiver and redraws the receiver.

`- takeColorFrom:` (page 765)
Changes the color of the receiver to that of the specified object.

Activating and Deactivating Color Wells

- `activate:` (page 762)
  Activates the receiver, displays the color panel, and makes the current color the same as its own.
- `deactivate` (page 763)
  Deactivates the receiver and redraws it.
- `isActive` (page 764)
  Returns a Boolean value indicating whether the receiver is active.

Managing Borders of Color Wells

- `isBordered` (page 764)
  Returns a Boolean value indicating whether the receiver has a border.
- `setBordered:` (page 764)
  Places or removes a border on the receiver and redraws the receiver.

Drawing a Color Well

- `drawWellInside:` (page 763)
  Draws the colored area inside the receiver at the specified location without drawing borders.

Instance Methods

activate:

Activates the receiver, displays the color panel, and makes the current color the same as its own.

```
- (void)activate:(BOOL)exclusive
```

Parameters

exclusive
`YES` to deactivate any other color wells; `NO` to keep them active. If a color panel is active with *exclusive* set to `YES` and another is subsequently activated with *exclusive* set to `NO`, the exclusive setting of the first panel is ignored.

Discussion
This method redraws the receiver. An active color well will have its color updated when the current color of the `NSColorPanel` changes. Any color well that shows its border highlights the border when it's active.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `deactivate` (page 763)
- `isActive` (page 764)

Declared In
NSColorWell.h

color

Returns the color of the receiver.

```
- (NSColor *)color
```

Return Value
The color of the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setColor: (page 765)
- takeColorFrom: (page 765)

Related Sample Code
CoreImageGLTextureFBO
DesktopImage
FunHouse
Grady
ObjectPath

Declared In
NSColorWell.h

deactivate

Deactivates the receiver and redraws it.

```
- (void)deactivate
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- activate: (page 762)
- isActive (page 764)

Related Sample Code
CWCocoaComponent

Declared In
NSColorWell.h

drawWellInside:

Draws the colored area inside the receiver at the specified location without drawing borders.

```
- (void)drawWellInside:(NSRect)insideRect
```

Parameters

insideRect

The rectangle specifying the area within which to draw.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSColorWell.h

isActive

Returns a Boolean value indicating whether the receiver is active.

```
- (BOOL)isActive
```

Return Value
YES if the receiver is active, NO otherwise.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSColorWell.h

isBordered

Returns a Boolean value indicating whether the receiver has a border.

```
- (BOOL)isBordered
```

Return Value
YES if the receiver is bordered, NO otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setBordered: (page 764)

Declared In
NSColorWell.h

setBordered:

Places or removes a border on the receiver and redraws the receiver.

```
- (void)setBordered:(BOOL)bordered
```

Parameters

bordered

YES to place a border on the receiver, NO to remove it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isBordered (page 764)

Declared In
NSColorWell.h

setColor:

Sets the color of the receiver and redraws the receiver.

```
- (void)setColor:(NSColor *)color
```

Parameters

color

The new color for the color well.

Availability
Available in Mac OS X v10.0 and later.

See Also
- color (page 763)
- takeColorFrom: (page 765)

Related Sample Code
FunHouse
Grady
Sketch-112

Declared In
NSColorWell.h

takeColorFrom:

Changes the color of the receiver to that of the specified object.

```
- (void)takeColorFrom:(id)sender
```

Parameters

sender

The object from which to take the new color.

Availability
Available in Mac OS X v10.0 and later.

See Also
- color (page 763)

- setColor: (page 765)

Declared In
NSColorWell.h

NSComboBox Class Reference

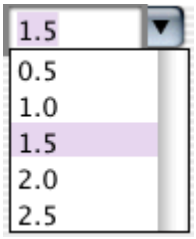
Inherits from	NSTextField : NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSTextField) NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSComboBox.h
Companion guide	Combo Box Programming Topics
Related sample code	DemoAssistant QTAudioExtractionPanel QTKitPlayer SimpleComboBox URL CacheInfo

Overview

An `NSComboBox` is a kind of `NSControl` that allows you to either enter text directly (as you would with an `NSTextField`) or click the attached arrow at the right of the combo box and select from a displayed ("pop-up") list of items.

Normally an instance of `NSComboBox` looks like this:

When you click the downward-pointing arrow at the right side of the text field, the pop-up list appears, like this:



The `NSComboBox` class uses `NSComboBoxCell` to implement its user interface.

Also see the `NSComboBoxDataSource` informal protocol, which declares the methods that an `NSComboBox` uses to access the contents of its data source object.

Tasks

Setting Display Attributes

- `hasVerticalScroller` (page 773)
  Returns a Boolean value indicating whether the receiver will display a vertical scroller.
- `intercellSpacing` (page 775)
  Returns the horizontal and vertical spacing between cells in the receiver's pop-up list.
- `isButtonBordered` (page 775)
  Returns whether the combo box button is set to display a border.
- `itemHeight` (page 775)
  Returns the height of each item in the receiver's pop-up list.
- `numberOfVisibleItems` (page 777)
  Returns the maximum number of items visible in the pop-up list.
- `setButtonBordered:` (page 782)
  Determines whether the button in the combo box is displayed with a border.
- `setHasVerticalScroller:` (page 783)
  Determines whether the receiver displays a vertical scroller.
- `setIntercellSpacing:` (page 784)
  Sets the spacing between pop-up list items.
- `setItemHeight:` (page 784)
  Sets the height for items.
- `setNumberOfVisibleItems:` (page 785)
  Sets the maximum number of items that are visible in the receiver's pop-up list.

Setting a Data Source

- `dataSource` (page 772)
  Returns the object that provides the data displayed in the receiver's pop-up list.

- `setDataSource:` (page 783)
  Sets the receiver's data source to *aSource*.
- `setUsesDataSource:` (page 785)
  Sets whether the receiver uses an external data source to populate the receiver's pop-up list.
- `usesDataSource` (page 786)
  Returns a Boolean value indicating whether the receiver uses an external data source to populate its pop-up list.

Working with an Internal List

- `addItemsWithObjectValues:` (page 770)
  Adds multiple objects to the end of the receiver's internal item list.
- `addItemWithObjectValue:` (page 771)
  Adds an object to the end of the receiver's internal item list.
- `insertItemWithObjectValue:atIndex:` (page 774)
  Inserts an object at the specified location in the receiver's internal item list.
- `objectValues` (page 778)
  Returns as an array the receiver's internal item list.
- `removeAllItems` (page 779)
  Removes all items from the receiver's internal item list.
- `removeItemAtIndex:` (page 779)
  Removes the object at the specified location from the receiver's internal item list.
- `removeItemWithObjectValue:` (page 779)
  Removes all occurrences of the given object from the receiver's internal item list.
- `numberOfItems` (page 777)
  Returns the total number of items in the pop-up list.

Manipulating the Displayed List

- `indexOfItemWithObjectValue:` (page 773)
  Searches the receiver's internal item list for the specified object and returns the lowest matching index.
- `itemObjectValueAtIndex:` (page 776)
  Returns the object located at the given index within the receiver's internal item list.
- `noteNumberOfItemsChanged` (page 776)
  Informs the receiver that the number of items in its data source has changed.
- `reloadData` (page 778)
  Marks the receiver as needing redisplay, so that it will reload the data for visible pop-up items and draw the new values.
- `scrollItemAtIndexToTop:` (page 780)
  Scrolls the receiver's pop-up list vertically so that the item at the specified index is as close to the top as possible.
- `scrollItemAtIndexToVisible:` (page 780)
  Scrolls the receiver's pop-up list vertically so that the item at the specified index is visible.

Manipulating the Selection

- `deselectItemAtIndex:` (page 772)
  Deselects the pop-up list item at the specified index if it's selected.
- `indexOfSelectedItem` (page 774)
  Returns the index of the last item selected from the pop-up list.
- `objectValueOfSelectedItem` (page 777)
  Returns the object corresponding to the last item selected from the pop-up list.
- `selectItemAtIndex:` (page 781)
  Selects the pop-up list row at the given index.
- `selectItemWithObjectValue:` (page 781)
  Selects the first pop-up list item that corresponds to the given object.

Completing the Text Field

- `completes` (page 771)
  Returns a Boolean value indicating whether the receiver tries to complete what the user types in the text field.
- `setCompletes:` (page 782)
  Sets whether the receiver tries to complete what the user types in the text field.

New Methods

- `delegate` (page 772)
  Returns the receiver's delegate.
- `setDelegate:` (page 783)
  Sets the receiver's delegate.

Instance Methods

addItemsWithObjectValues:

Adds multiple objects to the end of the receiver's internal item list.

```
- (void)addItemsWithObjectValues:(NSArray *)objects
```

Parameters

objects
  An array of the objects to add to the internal item list.

Discussion
This method logs a warning if `usesDataSource` (page 786) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
LSMSmartCategorizer
QTAudioExtractionPanel
QTKitPlayer

Declared In
NSComboBox.h

addItemWithObjectValue:

Adds an object to the end of the receiver's internal item list.

```
- (void)addItemWithObjectValue:(id)anObject
```

Parameters

anObject
The object to add to the internal item list.

Discussion
This method logs a warning if `usesDataSource` (page 786) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitPlayer

Declared In
NSComboBox.h

completes

Returns a Boolean value indicating whether the receiver tries to complete what the user types in the text field.

```
- (BOOL)completes
```

Return Value
`YES` if the receiver tries to complete what the user types in the text field; otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setCompletes:` (page 782)

Declared In
NSComboBox.h

dataSource

Returns the object that provides the data displayed in the receiver's pop-up list.

```
- (id < NSComboBoxDataSource >)dataSource
```

Return Value
The data source for the combo box's pop-up list.

Discussion
This method logs a warning if `usesDataSource` (page 786) returns `NO`. See the class description and the `NSComboBoxDataSource` informal protocol specification for more information on combo box data source objects.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSComboBox.h`

delegate

Returns the receiver's delegate.

```
- (id < NSComboBoxDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setDelegate:` (page 783)

Declared In
`NSComboBox.h`

deselectItemAtIndex:

Deselects the pop-up list item at the specified index if it's selected.

```
- (void)deselectItemAtIndex:(NSInteger)index
```

Parameters
index
The index of the item to deselect.

Discussion
If the selection does in fact change, this method posts an `NSComboBoxSelectionDidChangeNotification` (page 786) to the default notification center.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfSelectedItem` (page 774)
- `numberOfItems` (page 777)
- `selectItemAtIndex:` (page 781)

Declared In
`NSComboBox.h`

hasVerticalScroller

Returns a Boolean value indicating whether the receiver will display a vertical scroller.

```
- (BOOL)hasVerticalScroller
```

Return Value
`YES` if the receiver will display a vertical scroller; otherwise `NO`.

Discussion
Note that the scroller will be displayed even if the pop-up list contains fewer items than will fit in the area specified for display.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfItems` (page 777)
- `numberOfVisibleItems` (page 777)

Declared In
`NSComboBox.h`

indexOfItemWithObjectValue:

Searches the receiver's internal item list for the specified object and returns the lowest matching index.

```
- (NSInteger)indexOfItemWithObjectValue:(id)anObject
```

Parameters
anObject
The object for which to return the index.

Return Value
The lowest index in the internal item list whose corresponding value is equal to that of the specified object. Objects are considered equal if they have the same `id` or if `isEqual:` returns `YES`.

If none of the objects in the receiver's internal item list are equal to *anObject*, `indexOfItemWithObjectValue:` returns `NSNotFound`.

Discussion
This method logs a warning if `usesDataSource` (page 786) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectItemWithObjectValue:` (page 781)

Declared In
`NSComboBox.h`

indexOfSelectedItem

Returns the index of the last item selected from the pop-up list.

```
- (NSInteger)indexOfSelectedItem
```

Return Value
The index of the last item selected from the receiver's pop-up list or -1 if no item is selected.

Discussion
Note that nothing is initially selected in a newly initialized combo box.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `objectValueOfSelectedItem` (page 777)

Related Sample Code
ClipboardViewer

Declared In
`NSComboBox.h`

insertItemWithObjectValue:atIndex:

Inserts an object at the specified location in the receiver's internal item list.

```
- (void)insertItemWithObjectValue:(id)anObject atIndex:(NSInteger)index
```

Parameters

anObject
The object to add to the internal item list.

index
The index in the list at which to add the new object. The previous item at *index*—along with all following items—is shifted down one slot to make room

Discussion
This method logs a warning if `usesDataSource` (page 786) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addItemWithObjectValue:` (page 771)
- `numberOfItems` (page 777)

Related Sample Code
DemoAssistant

Declared In
`NSComboBox.h`

intercellSpacing

Returns the horizontal and vertical spacing between cells in the receiver's pop-up list.

```
- (NSSize)intercellSpacing
```

Return Value
The space between cells in the pop-up list. The default spacing is (3.0, 2.0).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `itemHeight` (page 775)
- `numberOfVisibleItems` (page 777)

Declared In
`NSComboBox.h`

isButtonBordered

Returns whether the combo box button is set to display a border.

```
- (BOOL)isButtonBordered
```

Return Value
`YES` if the button has a border; otherwise `NO`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setButtonBordered:` (page 782)

Declared In
`NSComboBox.h`

itemHeight

Returns the height of each item in the receiver's pop-up list.

```
- (CGFloat)itemHeight
```

Return Value
The height of items in the pop-up list. The default item height is 16.0.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `intercellSpacing` (page 775)
- `numberOfVisibleItems` (page 777)

Declared In
`NSComboBox.h`

itemObjectValueAtIndex:

Returns the object located at the given index within the receiver's internal item list.

```
- (id)itemObjectValueAtIndex:(NSInteger)index
```

Parameters

index

The index of the object to retrieve. If *index* is beyond the end of the list, an `NSRangeException` is raised.

Return Value
The object located at the specified index in the internal item list.

Discussion
This method logs a warning if `usesDataSource` (page 786) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `objectValueOfSelectedItem` (page 777)

Declared In
`NSComboBox.h`

noteNumberOfItemsChanged

Informs the receiver that the number of items in its data source has changed.

```
- (void)noteNumberOfItemsChanged
```

Discussion
This method allows the receiver to update the scrollers in its displayed pop-up list without actually reloading data into the receiver. It is particularly useful for a data source that continually receives data in the background over a period of time, in which case the `NSComboBox` can remain responsive to the user while the data is received.

See the `NSComboBoxDataSource` informal protocol specification for information on the messages an `NSComboBox` sends to its data source.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `reloadData` (page 778)

Declared In
`NSComboBox.h`

numberOfItems

Returns the total number of items in the pop-up list.

```
- (NSInteger)numberOfItems
```

Return Value
The number of items in the list.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfVisibleItems` (page 777)
- `numberOfItemsInComboBox:` (`NSComboBoxDataSource` protocol)

Related Sample Code
DemoAssistant

Declared In
`NSComboBox.h`

numberOfVisibleItems

Returns the maximum number of items visible in the pop-up list.

```
- (NSInteger)numberOfVisibleItems
```

Return Value
The maximum number of items visible at any one time in the pop-up list.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfItems` (page 777)

Declared In
`NSComboBox.h`

objectValueOfSelectedItem

Returns the object corresponding to the last item selected from the pop-up list.

```
- (id)objectValueOfSelectedItem
```

Return Value
The object in the receiver's internal item list corresponding to the last item selected from the pop-up list, or `nil` if no item is selected.

Discussion
Note that nothing is initially selected in a newly initialized combo box. This method logs a warning if `usesDataSource` (page 786) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- indexOfSelectedItem` (page 774)
`- comboBox:objectValueForItemAtIndex:` (NSComboBoxDataSource protocol)

Related Sample Code
QTAudioExtractionPanel
QTKitPlayer

Declared In
`NSComboBox.h`

objectValues

Returns as an array the receiver's internal item list.

```
- (NSArray *)objectValues
```

Return Value
The array containing the objects in the receiver's internal item list.

Discussion
This method logs a warning if `usesDataSource` (page 786) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DemoAssistant

Declared In
`NSComboBox.h`

reloadData

Marks the receiver as needing redisplay, so that it will reload the data for visible pop-up items and draw the new values.

```
- (void)reloadData
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- noteNumberOfItemsChanged (page 776)

Declared In
NSComboBox.h

removeAllItems

Removes all items from the receiver's internal item list.

```
- (void)removeAllItems
```

Discussion
This method logs a warning if usesDataSource (page 786) returns YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- objectValues (page 778)

Declared In
NSComboBox.h

removeItemAtIndex:

Removes the object at the specified location from the receiver's internal item list.

```
- (void)removeItemAtIndex:(NSInteger)index
```

Parameters
index
The index of the object to remove. All items beyond *index* are moved up one slot to fill the gap.

Discussion
The removed object receives a release message. This method raises an NSRangeException if *index* is beyond the end of the list and logs a warning if usesDataSource (page 786) returns YES.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DemoAssistant
QTAudioExtractionPanel
QTKitPlayer

Declared In
NSComboBox.h

removeItemWithObjectValue:

Removes all occurrences of the given object from the receiver's internal item list.

```
- (void)removeItemWithObjectValue:(id)anObject
```

Parameters

anObject

The object to remove from the internal item list. Objects are considered equal if they have the same `id` or if `isEqual:` returns `YES`.

Discussion

This method logs a warning if `usesDataSource` (page 786) returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- indexOfItemWithObjectValue:` (page 773)

Related Sample Code

QTAudioExtractionPanel

QTKitPlayer

Declared In

`NSComboBox.h`

scrollItemAtIndexToTop:

Scrolls the receiver's pop-up list vertically so that the item at the specified index is as close to the top as possible.

```
- (void)scrollItemAtIndexToTop:(NSInteger)index
```

Parameters

index

The index of the item to scroll to the top.

Discussion

The pop-up list need not be displayed at the time this method is invoked.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSComboBox.h`

scrollItemAtIndexToVisible:

Scrolls the receiver's pop-up list vertically so that the item at the specified index is visible.

```
- (void)scrollItemAtIndexToVisible:(NSInteger)index
```

Parameters

index

The index of the item to make visible.

Discussion
The pop-up list need not be displayed at the time this method is invoked.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSComboBox.h

selectItemAtIndex:

Selects the pop-up list row at the given index.

```
- (void)selectItemAtIndex:(NSInteger)index
```

Parameters

index
The index of the item to select in the pop-up list.

Discussion
Posts an NSComboBoxSelectionDidChangeNotification (page 786) to the default notification center if the selection does in fact change. Note that this method does not alter the contents of the combo box's text field—see Setting the Combo Box's Value for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setObjectValue: (page 836) (NSControl)

Declared In
NSComboBox.h

selectItemWithObjectValue:

Selects the first pop-up list item that corresponds to the given object.

```
- (void)selectItemWithObjectValue:(id)anObject
```

Parameters

anObject
The object to select in the pop-up list. Objects are considered equal if they have the same id or if isEqual: returns YES.

Discussion
This method logs a warning if usesDataSource (page 786) returns YES. Posts an NSComboBoxSelectionDidChangeNotification (page 786) to the default notification center if the selection does in fact change. Note that this method doesn't alter the contents of the combo box's text field—see Setting the Combo Box's Value for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setObjectValue:` (page 836) (NSControl)

Declared In
`NSComboBox.h`

setButtonBordered:

Determines whether the button in the combo box is displayed with a border.

```
- (void)setButtonBordered:(BOOL)flag
```

Parameters

flag

`YES` to display a border; `NO` to display the button without a border. For example, it is often useful when using a combo box in an `NSTableView` to display the button without the border.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `isButtonBordered` (page 775)

Declared In
`NSComboBox.h`

setCompletes:

Sets whether the receiver tries to complete what the user types in the text field.

```
- (void)setCompletes:(BOOL)completes
```

Parameters

completes

`YES` to indicate that the receiver should try to complete test entered by the user. If *completes* is `YES`, every time the user adds characters to the end of the text field, the combo box calls the `NSComboBoxCell` method `completedString:` (page 792).

Discussion
If `completedString:` (page 792) returns a string that's longer than the existing string, the combo box replaces the existing string with the returned string and selects the additional characters. If the user is deleting characters or adds characters somewhere besides the end of the string, the combo box does not try to complete it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `completes` (page 771)

Declared In
`NSComboBox.h`

setDataSource:

Sets the receiver's data source to *aSource*.

```
- (void)setDataSource:(id < NSComboBoxDataSource >)aSource
```

Parameters

aSource

The new data source for the receiver. The data source should implement the appropriate methods of the `NSComboBoxDataSource` informal protocol.

This method logs a warning if *aSource* doesn't respond to either `numberOfItemsInComboBox:` or `comboBox:objectValueForItemAtIndex:`.

Discussion

This method doesn't automatically set `usesDataSource` (page 786) to `NO` and in fact logs a warning if `usesDataSource` (page 786) returns `NO`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setUsesDataSource:` (page 785)

Declared In

`NSComboBox.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSComboBoxDelegate >)anObject
```

Parameters

anObject

The delegate for the receiver. The delegate must conform to the `NSComboBoxDelegate Protocol` protocol.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- delegate` (page 772)

Declared In

`NSComboBox.h`

setHasVerticalScroller:

Determines whether the receiver displays a vertical scroller.

```
- (void)setHasVerticalScroller:(BOOL)flag
```

Parameters

flag

YES to display a vertical scroller; NO otherwise. By default, *flag* is YES.

Discussion

If *flag* is NO and the combo box has more list items (either in its internal item list or from its data source) than are allowed by numberOfVisibleItems (page 777), only a subset are displayed. The NSComboBox class' scroll... methods can be used to position this subset within the pop-up list.

Note that if flag is YES, a scroller will be displayed even if the combo box has fewer list items than are allowed by numberOfVisibleItems (page 777).

Availability

Available in Mac OS X v10.0 and later.

See Also

- numberOfItems (page 777)
- scrollItemAtIndexToTop: (page 780)
- scrollItemAtIndexToVisible: (page 780)

Declared In

NSComboBox.h

setIntercellSpacing:

Sets the spacing between pop-up list items.

```
- (void)setIntercellSpacing:(NSSize)aSize
```

Parameters

aSize

The new width and height between pop-up list items. The default intercell spacing is (3.0, 2.0).

Availability

Available in Mac OS X v10.0 and later.

See Also

- setItemHeight: (page 784)
- setNumberOfVisibleItems: (page 785)

Declared In

NSComboBox.h

setItemHeight:

Sets the height for items.

```
- (void)setItemHeight:(CGFloat)itemHeight
```

Parameters

itemHeight

The new height for items in the pop-up list.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setIntercellSpacing:` (page 784)
- `setNumberOfVisibleItems:` (page 785)

Declared In
`NSComboBox.h`

setNumberOfVisibleItems:

Sets the maximum number of items that are visible in the receiver's pop-up list.

```
- (void)setNumberOfVisibleItems:(NSInteger)visibleItems
```

Parameters

visibleItems
The maximum number of items that are visible at one time in the pop-up list.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfItems` (page 777)
- `setItemHeight:` (page 784)
- `setIntercellSpacing:` (page 784)

Declared In
`NSComboBox.h`

setUsesDataSource:

Sets whether the receiver uses an external data source to populate the receiver's pop-up list.

```
- (void)setUsesDataSource:(BOOL)flag
```

Parameters

flag
`YES` if the receiver uses an external data source (specified by `setDataSource:` (page 783)); otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSComboBox.h`

usesDataSource

Returns a Boolean value indicating whether the receiver uses an external data source to populate its pop-up list.

```
- (BOOL)usesDataSource
```

Return Value
`YES` if the receiver uses an external data source to populate the receiver's pop-up list, `NO` if it uses an internal item list.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- dataSource` (page 772)

Declared In
`NSComboBox.h`

Notifications

NSComboBoxSelectionDidChangeNotification

Posted after the pop-up list selection of the `NSComboBox` changes.

The notification object is the `NSComboBox` whose selection changed. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSComboBox.h`

NSComboBoxSelectionIsChangingNotification

Posted whenever the pop-up list selection of the `NSComboBox` is changing.

The notification object is the `NSComboBox` whose selection is changing. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSComboBox.h`

NSComboBoxWillDismissNotification

Posted whenever the pop-up list of the `NSComboBox` is about to be dismissed.

The notification object is the `NSComboBox` whose pop-up list will be dismissed. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSComboBox.h`

NSComboBoxWillPopUpNotification

Posted whenever the pop-up list of the `NSComboBox` is going to be displayed.

The notification object is the `NSComboBox` whose pop-up window will be displayed. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSComboBox.h`

NSComboBoxCell Class Reference

Inherits from	NSTextFieldCell : NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSComboBoxCell.h
Companion guide	Combo Box Programming Topics
Related sample code	SimpleComboBox

Overview

`NSComboBoxCell` is a subclass of `NSTextFieldCell` used to implement the user interface of "combo boxes" (see `NSComboBox` for information on how combo boxes look and work). The `NSComboBox` subclass of `NSTextField` uses a single `NSComboBoxCell`, and essentially all of the `NSComboBox` class' methods simply invoke the corresponding `NSComboBoxCell` method.

Also see the `NSComboBoxCellDataSource` protocol, which declares the methods that an `NSComboBoxCell` object uses to access the contents of its data source object.

Tasks

Setting Display Attributes

- `hasVerticalScroller` (page 794)
  Returns a Boolean value indicating whether the receiver will display a vertical scroller.
- `isButtonBordered` (page 796)
  Returns a Boolean value indicating whether the combo box button is set to display a border.
- `intercellSpacing` (page 796)
  Returns the spacing between cells in the receiver's pop-up list.
- `itemHeight` (page 797)
  Returns the height of each item in the receiver's pop-up list.

- `numberOfVisibleItems` (page 798)
  Returns the maximum number of items visible in the pop-up list.
- `setButtonBordered:` (page 803)
  Determines whether the button in the combo box is displayed with a border.
- `setHasVerticalScroller:` (page 804)
  Determines whether the receiver displays a vertical scroller.
- `setIntercellSpacing:` (page 805)
  Sets the spacing between pop-up list items.
- `setItemHeight:` (page 805)
  Sets the height for items.
- `setNumberOfVisibleItems:` (page 805)
  Sets the maximum number of items that are visible in the pop-up list.

Setting a Data Source

- `dataSource` (page 793)
  Returns the object that provides the data displayed in the receiver's pop-up list.
- `setDataSource:` (page 804)
  Sets the receiver's data source.
- `setUsesDataSource:` (page 806)
  Sets whether the receiver uses an external data source to populate its pop-up list.
- `usesDataSource` (page 806)
  Returns a Boolean value indicating whether the receiver uses an external data source.

Working with an Internal List

- `addItemsWithObjectValues:` (page 792)
  Adds multiple objects to the internal item list.
- `addItemWithObjectValue:` (page 792)
  Adds the specified object to the internal item list.
- `insertItemWithObjectValue:atIndex:` (page 795)
  Inserts an object at the specified location in the internal item list.
- `objectValues` (page 799)
  Returns the receiver's internal item list.
- `removeAllItems` (page 800)
  Removes all items from the receiver's internal item list.
- `removeItemAtIndex:` (page 800)
  Removes the object at the specified location from the receiver's internal item list.
- `removeItemWithObjectValue:` (page 801)
  Removes all occurrences of the specified object from the receiver's internal item list.
- `numberOfItems` (page 798)
  Returns the total number of items in the pop-up list.

Manipulating the Displayed List

- `indexOfItemWithObjectValue:` (page 795)
  Searches the receiver's internal item list for the given object and returns the matching index number.
- `itemObjectValueAtIndex:` (page 797)
  Returns the object located at the specified location in the internal item list.
- `noteNumberOfItemsChanged` (page 798)
  Informs the receiver that the number of items in its data source has changed.
- `reloadData` (page 799)
  Marks the receiver as needing redisplay, so that it will reload the data for visible pop-up items and draw the new values.
- `scrollItemAtIndexToTop:` (page 801)
  Scrolls the receiver's pop-up list vertically so that the item at the given index is as close to the top as possible.
- `scrollItemAtIndexToVisible:` (page 801)
  Scrolls the receiver's pop-up list vertically so that the item at the given index is visible.

Manipulating the Selection

- `deselectItemAtIndex:` (page 794)
  Deselects the pop-up list item at the given index if it's selected.
- `indexOfSelectedItem` (page 795)
  Returns the index of the last item selected from the pop-up list.
- `objectValueOfSelectedItem` (page 799)
  Returns the object corresponding to the last item selected from the pop-up list.
- `selectItemAtIndex:` (page 802)
  Selects the pop-up list row at the given index.
- `selectItemWithObjectValue:` (page 802)
  Selects the first pop-up list item that corresponds to the specified object.

Completing the Text Field

- `completedString:` (page 792)
  Returns a string from the receiver's pop-up list that starts with the given substring.
- `completes` (page 793)
  Returns a Boolean value indicating whether the receiver tries to complete text entered by the user.
- `setCompletes:` (page 803)
  Sets whether the receiver tries to complete what the user types in the text field.

Instance Methods

addItemsWithObjectValues:

Adds multiple objects to the internal item list.

```
- (void)addItemsWithObjectValues:(NSArray *)objects
```

Parameters

objects

The object to add to the end of the receiver's internal item list.

Discussion

This method logs a warning if `usesDataSource` (page 806) returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSComboBoxCell.h`

addItemWithObjectValue:

Adds the specified object to the internal item list.

```
- (void)addItemWithObjectValue:(id)anObject
```

Parameters

anObject

The object to add to the end of the receiver's internal item list.

Discussion

This method logs a warning if `usesDataSource` (page 806) returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSComboBoxCell.h`

completedString:

Returns a string from the receiver's pop-up list that starts with the given substring.

```
- (NSString *)completedString:(NSString *)substring
```

Parameters

substring

The substring to search for. This is what the user entered in the combo box's text field.

Return Value
The string from the receiver's pop-up list that starts with the specified substring or `nil` if there is no such string.

Discussion
The default implementation of this method first checks whether the combo box uses a data source and whether the data source responds to `comboBox:completedString:` or `comboBoxCell:completedString:`. If so, the combo box cell returns that method's return value. Otherwise, this method goes through the combo box's items one by one and returns an item that starts with *`substring`*.

Override this method only if your subclass completes strings differently. The overriding method does not need to call the superclass's method. Generally, you do not need to call this method directly.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSComboBoxCell.h`

completes

Returns a Boolean value indicating whether the receiver tries to complete text entered by the user.

```
- (BOOL)completes
```

Return Value
`YES` if the receiver tries to complete what the user types in the text field; `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setCompletes:` (page 803)

Declared In
`NSComboBoxCell.h`

dataSource

Returns the object that provides the data displayed in the receiver's pop-up list.

```
- (id < NSComboBoxCellDataSource >)dataSource
```

Return Value
The data source for the receiver's pop-up list.

Discussion
This method logs a warning if `usesDataSource` (page 806) returns `NO`. See the class description and the `NSComboBoxCellDataSource` informal protocol specification for more information on combo box cell data source objects.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSComboBoxCell.h

deselectItemAtIndex:

Deselects the pop-up list item at the given index if it's selected.

```
- (void)deselectItemAtIndex:(NSInteger)index
```

Parameters

index
The index of the item to deselect.

Discussion
If the selection does in fact change, this method posts an `NSComboBoxSelectionDidChangeNotification` (page 786) to the default notification center.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfSelectedItem` (page 795)
- `numberOfItems` (page 798)
- `selectItemAtIndex:` (page 802)

Declared In
NSComboBoxCell.h

hasVerticalScroller

Returns a Boolean value indicating whether the receiver will display a vertical scroller.

```
- (BOOL)hasVerticalScroller
```

Return Value
`YES` if the receiver displays a vertical scroller; otherwise `NO`.

Discussion
Note that the scroller will be displayed even if the pop-up list contains fewer items than will fit in the area specified for display.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfItems` (page 798)
- `numberOfVisibleItems` (page 798)

Declared In
NSComboBoxCell.h

indexOfItemWithObjectValue:

Searches the receiver's internal item list for the given object and returns the matching index number.

```
- (NSInteger)indexOfItemWithObjectValue:(id)anObject
```

Parameters

anObject

The object for which to return the index.

Return Value

The lowest index whose corresponding value is equal to *anObject*. Objects are considered equal if they have the same `id` or if `isEqual:` returns `YES`. If none of the objects in the receiver's internal item list is equal to *anObject*, `indexOfItemWithObjectValue:` returns `NSNotFound`.

Discussion

This method logs a warning if `usesDataSource` (page 806) returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- selectItemWithObjectValue:` (page 802)

Declared In

`NSComboBoxCell.h`

indexOfSelectedItem

Returns the index of the last item selected from the pop-up list.

```
- (NSInteger)indexOfSelectedItem
```

Return Value

The index of the last item selected from the receiver's pop-up list or –1 if no item is selected.

Discussion

Note that nothing is initially selected in a newly initialized combo box cell.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- objectValueOfSelectedItem` (page 799)

Declared In

`NSComboBoxCell.h`

insertItemWithObjectValue:atIndex:

Inserts an object at the specified location in the internal item list.

```
- (void)insertItemWithObjectValue:(id)anObject atIndex:(NSInteger)index
```

Parameters

anObject

The object to add to the receiver's internal item list.

index

The index at which to add the specified object. The previous item at *index*—along with all following items—is shifted down one slot to make room.

Discussion

This method logs a warning if `usesDataSource` (page 806) returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addItemWithObjectValue:` (page 792)
- `numberOfItems` (page 798)

Declared In

`NSComboBoxCell.h`

intercellSpacing

Returns the spacing between cells in the receiver's pop-up list.

```
- (NSSize)intercellSpacing
```

Return Value

The horizontal and vertical spacing between cells in the receiver's pop-up list. The default spacing is (3.0, 2.0).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `itemHeight` (page 797)
- `numberOfVisibleItems` (page 798)

Declared In

`NSComboBoxCell.h`

isButtonBordered

Returns a Boolean value indicating whether the combo box button is set to display a border.

```
- (BOOL)isButtonBordered
```

Return Value

`YES` if the button has a border; otherwise `NO`.

Availability

Available in Mac OS X v10.3 and later.

See Also
- `setButtonBordered:` (page 803)

Declared In
`NSComboBoxCell.h`

itemHeight

Returns the height of each item in the receiver's pop-up list.

```
- (CGFloat)itemHeight
```

Return Value
The height of each item in the pop-up list. The default item height is 16.0.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `intercellSpacing` (page 796)
- `numberOfVisibleItems` (page 798)

Declared In
`NSComboBoxCell.h`

itemObjectValueAtIndex:

Returns the object located at the specified location in the internal item list.

```
- (id)itemObjectValueAtIndex:(NSInteger)index
```

Parameters
index
The index of the object to return. If *index* is beyond the end of the list, an `NSRangeException` is raised.

Return Value
The object at the given location in the receiver's internal item list.

Discussion
This method logs a warning if `usesDataSource` (page 806) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `objectValueOfSelectedItem` (page 799)

Declared In
`NSComboBoxCell.h`

noteNumberOfItemsChanged

Informs the receiver that the number of items in its data source has changed.

```
- (void)noteNumberOfItemsChanged
```

Discussion
This method allows the receiver to update the scrollers in its displayed pop-up list without actually reloading data into the receiver. It is particularly useful for a data source that continually receives data in the background over a period of time, in which case the `NSComboBoxCell` can remain responsive to the user while the data is received.

See the `NSComboBoxCellDataSource` informal protocol specification for information on the messages an `NSComboBoxCell` sends to its data source.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `reloadData` (page 799)

Declared In
`NSComboBoxCell.h`

numberOfItems

Returns the total number of items in the pop-up list.

```
- (NSInteger)numberOfItems
```

Return Value
The number of items in the pop-up list.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfVisibleItems` (page 798)
- `numberOfItemsInComboBoxCell:` (NSComboBoxCellDataSource protocol)

Declared In
`NSComboBoxCell.h`

numberOfVisibleItems

Returns the maximum number of items visible in the pop-up list.

```
- (NSInteger)numberOfVisibleItems
```

Return Value
The maximum number of items that are visible in the receiver's pop-up list at any one time.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfItems` (page 798)

Declared In
`NSComboBoxCell.h`

objectValueOfSelectedItem

Returns the object corresponding to the last item selected from the pop-up list.

```
- (id)objectValueOfSelectedItem
```

Return Value
The object from the receiver's internal item list corresponding to the last item selected from the pop-up list, or `nil` if no item is selected.

Discussion
Note that nothing is initially selected in a newly initialized combo box cell. This method logs a warning if `usesDataSource` (page 806) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfSelectedItem` (page 795)
- `comboBoxCell:objectValueForItemAtIndex:` (`NSComboBoxCellDataSource` protocol)

Declared In
`NSComboBoxCell.h`

objectValues

Returns the receiver's internal item list.

```
- (NSArray *)objectValues
```

Return Value
An array containing the objects in the receiver's internal item list.

Discussion
This method logs a warning if `usesDataSource` (page 806) returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSComboBoxCell.h`

reloadData

Marks the receiver as needing redisplay, so that it will reload the data for visible pop-up items and draw the new values.

```
- (void)reloadData
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- noteNumberOfItemsChanged (page 798)

Declared In
NSComboBoxCell.h

removeAllItems

Removes all items from the receiver's internal item list.

```
- (void)removeAllItems
```

Discussion
This method logs a warning if usesDataSource (page 806) returns YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- objectValues (page 799)

Declared In
NSComboBoxCell.h

removeItemAtIndex:

Removes the object at the specified location from the receiver's internal item list.

```
- (void)removeItemAtIndex:(NSInteger)index
```

Parameters
index
The index of the object to remove from the receiver's internal item list. All items beyond *index* are moved up one slot to fill the gap.

Discussion
The removed object receives a release message. This method raises an NSRangeException if *index* is beyond the end of the list and logs a warning if usesDataSource (page 806) returns YES.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSComboBoxCell.h

removeItemWithObjectValue:

Removes all occurrences of the specified object from the receiver's internal item list.

```
- (void)removeItemWithObjectValue:(id)anObject
```

Parameters

anObject

The object to remove from the receiver's internal item list. Objects are considered equal if they have the same `id` or if `isEqual:` returns `YES`.

Discussion

This method logs a warning if `usesDataSource` (page 806) returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- indexOfItemWithObjectValue:` (page 795)

Declared In

`NSComboBoxCell.h`

scrollItemAtIndexToTop:

Scrolls the receiver's pop-up list vertically so that the item at the given index is as close to the top as possible.

```
- (void)scrollItemAtIndexToTop:(NSInteger)index
```

Parameters

index

The index of the item to scroll to the top.

Discussion

The pop-up list need not be displayed at the time this method is invoked.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSComboBoxCell.h`

scrollItemAtIndexToVisible:

Scrolls the receiver's pop-up list vertically so that the item at the given index is visible.

```
- (void)scrollItemAtIndexToVisible:(NSInteger)index
```

Parameters

index

The index of the item to make visible.

Discussion

The pop-up list need not be displayed at the time this method is invoked.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSComboBoxCell.h

selectItemAtIndex:

Selects the pop-up list row at the given index.

```
- (void)selectItemAtIndex:(NSInteger)index
```

Parameters

index

The index of the row to select.

Discussion
Posts an NSComboBoxSelectionDidChangeNotification (page 786) to the default notification center if the selection does in fact change. Note that this method does not alter the contents of the combo box cell's text field—see Setting the Combo Box's Value for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setObjectValue: (page 836) (NSControl)

Declared In
NSComboBoxCell.h

selectItemWithObjectValue:

Selects the first pop-up list item that corresponds to the specified object.

```
- (void)selectItemWithObjectValue:(id)anObject
```

Parameters

anObject

The object for which to select the corresponding pop-up list item. Objects are considered equal if they have the same id or if isEqual: returns YES.

Discussion
This method logs a warning if usesDataSource (page 806) returns YES. Posts an NSComboBoxSelectionDidChangeNotification (page 786) to the default notification center if the selection does in fact change. Note that this method doesn't alter the contents of the combo box cell's text field—see Setting the Combo Box's Value for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setObjectValue: (page 836) (NSControl)

Declared In
NSComboBoxCell.h

setButtonBordered:

Determines whether the button in the combo box is displayed with a border.

```
- (void)setButtonBordered:(BOOL)flag
```

Parameters

flag

YES to display a border. For example, it is often useful when using a combo box in an NSTableView to display the button without the border.

Availability
Available in Mac OS X v10.3 and later.

See Also
- isButtonBordered (page 796)

Declared In
NSComboBoxCell.h

setCompletes:

Sets whether the receiver tries to complete what the user types in the text field.

```
- (void)setCompletes:(BOOL)completes
```

Parameters

completes

YES to indicate that the receiver should try to complete text typed by the user. If *completes* is YES, every time the user adds characters to the end of the text field, the combo box calls the NSComboBoxCell method completedString: (page 792).

Discussion
If completedString: (page 792) returns a string that's longer than the existing string, the combo box replaces the existing string with the returned string and selects the additional characters. If the user is deleting characters or adds characters somewhere besides the end of the string, the combo box does not try to complete it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- completes (page 793)

Declared In
NSComboBoxCell.h

setDataSource:

Sets the receiver's data source.

```
- (void)setDataSource:(id < NSComboBoxCellDataSource >)aSource
```

Parameters

aSource

The data source for the receiver. *aSource* should implement the appropriate methods of the NSComboBoxCellDataSource informal protocol.

This method logs a warning if *aSource* doesn't respond to either numberOfItemsInComboBoxCell: or comboBoxCell:objectValueForItemAtIndex:.

Discussion

This method doesn't automatically set usesDataSource (page 806) to NO and in fact logs a warning if usesDataSource returns NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setUsesDataSource: (page 806)

Declared In

NSComboBoxCell.h

setHasVerticalScroller:

Determines whether the receiver displays a vertical scroller.

```
- (void)setHasVerticalScroller:(BOOL)flag
```

Parameters

flag

YES to have the receiver display a vertical scroller. By default, *flag* is YES.

Discussion

If *flag* is NO and the combo box cell has more list items (either in its internal item list or from its data source) than are allowed by numberOfVisibleItems (page 798), only a subset will be displayed. NSComboBoxCell's scroll... methods can be used to position this subset within the pop-up list.

Note that if *flag* is YES, a scroller will be displayed even if the combo box cell has fewer list items than are allowed by numberOfVisibleItems.

Availability

Available in Mac OS X v10.0 and later.

See Also

- numberOfItems (page 798)
- scrollItemAtIndexToTop: (page 801)
- scrollItemAtIndexToVisible: (page 801)

Declared In

NSComboBoxCell.h

setIntercellSpacing:

Sets the spacing between pop-up list items.

```
- (void)setIntercellSpacing:(NSSize)aSize
```

Parameters

aSize

The width and height between pop-up list items. The default intercell spacing is (3.0, 2.0).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setItemHeight:` (page 805)
- `setNumberOfVisibleItems:` (page 805)

Declared In
`NSComboBoxCell.h`

setItemHeight:

Sets the height for items.

```
- (void)setItemHeight:(CGFloat)itemHeight
```

Parameters

itemHeight

The height of pop-up list items.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setIntercellSpacing:` (page 805)
- `setNumberOfVisibleItems:` (page 805)

Declared In
`NSComboBoxCell.h`

setNumberOfVisibleItems:

Sets the maximum number of items that are visible in the pop-up list.

```
- (void)setNumberOfVisibleItems:(NSInteger)visibleItems
```

Parameters

visibleItems

The maximum number of items that should be visible at one time in the receiver's pop-up list.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfItems` (page 798)
- `numberOfVisibleItems` (page 798)
- `setIntercellSpacing:` (page 805)
- `setItemHeight:` (page 805)

Declared In
`NSComboBoxCell.h`

setUsesDataSource:

Sets whether the receiver uses an external data source to populate its pop-up list.

```
- (void)setUsesDataSource:(BOOL)flag
```

Parameters

flag

`YES` to indicate that the receiver uses an external data source (specified by `setDataSource:` (page 804)) to populate the receiver's pop-up list.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSComboBoxCell.h`

usesDataSource

Returns a Boolean value indicating whether the receiver uses an external data source.

```
- (BOOL)usesDataSource
```

Return Value
`YES` if the receiver uses an external data source to populate the receiver's pop-up list, `NO` if it uses an internal item list.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `dataSource` (page 793)

Declared In
`NSComboBoxCell.h`

NSControl Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSControl.h
Companion guide	Control and Cell Programming Topics for Cocoa
Related sample code	ClockControl EnhancedAudioBurn ImageClient QTMetadataEditor Quartz Composer QCTV

Overview

`NSControl` is an abstract superclass that provides three fundamental features for implementing user interface devices: drawing devices on the screen, responding to user events, and sending action messages. It works closely with the `NSCell` class.

About Delegate Methods

The `NSControl` class provides several delegate methods for its subclasses that allow text editing, such as `NSTextField` and `NSMatrix`. These include: `controlTextDidBeginEditing:` (page 846), `controlTextDidChange:` (page 846), and `controlTextDidEndEditing:` (page 847).

Note that although `NSControl` defines delegate methods, it does not itself have a delegate. Any subclass that uses these methods must have a delegate and the methods to get and set it. In addition, a formal delegate protocol `NSControlTextEditingDelegate` also defines delegate methods used by control delegates.

Tasks

Initializing an NSControl

- `initWithFrame:` (page 820)
  Returns an NSControl object initialized with the specified frame rectangle.

Setting the Control's Cell

+ `cellClass` (page 812)
  Returns the type of cell used by the receiver.
+ `setCellClass:` (page 813)
  Sets the type of cell used by the receiver.
- `cell` (page 816)
  Returns the receiver's cell object.
- `setCell:` (page 830)
  Sets the receiver's cell

Enabling and Disabling the Control

- `isEnabled` (page 823)
  Returns whether the receiver reacts to mouse events.
- `setEnabled:` (page 831)
  Sets whether the receiver (and its cell) reacts to mouse events.

Identifying the Selected Cell

- `selectedCell` (page 825)
  Returns the receiver's selected cell.
- `selectedTag` (page 826)
  Returns the tag of the receiver's selected cell.

Setting the Control's Value

- `doubleValue` (page 817)
  Returns the value of the receiver's cell as a double-precision floating-point number.
- `setDoubleValue:` (page 831)
  Sets the value of the receiver's cell using a double-precision floating-point number.
- `floatValue` (page 819)
  Returns the value of the receiver's cell as a single-precision floating-point number.

- `setFloatValue:` (page 832)
  Sets the value of the receiver's cell using a single-precision floating-point number.
- `intValue` (page 821)
  Returns the value of the receiver's cell as an integer.
- `setIntValue:` (page 835)
  Sets the value of the receiver's cell using an integer.
- `integerValue` (page 821)
  Returns the value of the receiver's cell as an `NSInteger` value.
- `setIntegerValue:` (page 834)
  Sets the value of the receiver's cell using an `NSInteger` value.
- `objectValue` (page 824)
  Returns the value of the receiver's cell as an Objective-C object.
- `setObjectValue:` (page 836)
  Sets the value of the receiver's cell using an Objective-C object.
- `stringValue` (page 840)
  Returns the value of the receiver's cell as an `NSString` object.
- `setStringValue:` (page 837)
  Sets the value of the receiver's cell using an `NSString` object.
- `setNeedsDisplay` (page 836)
  Marks the receiver as needing redisplay (assuming automatic display is enabled).
- `attributedStringValue` (page 815)
  Returns the value of the receiver's cell as an attributed string.
- `setAttributedStringValue:` (page 829)
  Sets the value of the receiver's cell using an attributed string.

Interacting with Other Controls

- `takeDoubleValueFrom:` (page 841)
  Sets the value of the receiver's cell to a double-precision floating-point value obtained from the specified object.
- `takeFloatValueFrom:` (page 841)
  Sets the value of the receiver's cell to a single-precision floating-point value obtained from the specified object.
- `takeIntValueFrom:` (page 842)
  Sets the value of the receiver's cell to an integer value obtained from the specified object.
- `takeIntegerValueFrom:` (page 842)
  Sets the value of the receiver's cell to an `NSInteger` value obtained from the specified object.
- `takeObjectValueFrom:` (page 843)
  Sets the value of the receiver's cell to the object value obtained from the specified object.
- `takeStringValueFrom:` (page 843)
  Sets the value of the receiver's cell to the string value obtained from the specified object.

Formatting Text

- `alignment` (page 814)
  Returns the alignment mode of the text in the receiver's cell.
- `setAlignment:` (page 828)
  Sets the alignment of text in the receiver's cell.
- `font` (page 819)
  Returns the font used to draw text in the receiver's cell.
- `setFont:` (page 833)
  Sets the font used to draw text in the receiver's cell.
- `formatter` (page 820)
  Returns the receiver's formatter.
- `setFormatter:` (page 834)
  Sets the receiver's formatter
- `baseWritingDirection` (page 815)
  Returns the initial writing direction used to determine the actual writing direction for text.
- `setBaseWritingDirection:` (page 829)
  Sets the initial writing direction used to determine the actual writing direction for text .
- `setFloatingPointFormat:left:right:` (page 832) Deprecated in Mac OS X v10.0 Available in Mac OS X v10.0 through Mac OS X v10.5
  Sets the auto-ranging and floating point number format of the receiver's cell.

Managing the Field Editor

- `abortEditing` (page 813)
  Terminates the current editing operation and discards any edited text.
- `currentEditor` (page 817)
  Returns the current field editor for the control.
- `validateEditing` (page 845)
  Validates changes to any user-typed text.

Resizing the Control

- `calcSize` (page 816)
  Recomputes any internal sizing information for the receiver, if necessary.
- `sizeToFit` (page 839)
  Resizes the receiver's frame so that it is the minimum size needed to contain its cell.

Displaying a Cell

- `selectCell:` (page 825)
  Selects the specified cell and redraws the control as needed.

- `drawCell:` (page 818)
  Draws the specified cell, as long as it belongs to the receiver.
- `drawCellInside:` (page 818)
  Draws the inside of the receiver's cell (the area within the bezel or border)
- `updateCell:` (page 844)
  Marks the specified cell as in need of redrawing.
- `updateCellInside:` (page 845)
  Marks the inside of the specified cell as in need of redrawing.

Implementing the Target/action Mechanism

- `action` (page 814)
  Returns the default action-message selector associated with the control.
- `setAction:` (page 828)
  Sets the receiver's action method to the specified selector.
- `target` (page 844)
  Returns the target object of the receiver's cell.
- `setTarget:` (page 838)
  Sets the target object to receive action messages from the receiver's cell.
- `isContinuous` (page 822)
  Returns a Boolean value indicating whether the receiver's cell sends its action message continuously to its target during mouse tracking.
- `setContinuous:` (page 830)
  Sets whether the receiver's cell sends its action message continuously to its target during mouse tracking.
- `sendAction:to:` (page 826)
  Causes the specified action to be sent the target.
- `sendActionOn:` (page 827)
  Sets the conditions on which the receiver sends action messages to its target.

Getting and Setting Tags

- `tag` (page 840)
  Returns the tag identifying the receiver (not the tag of the receiver's cell).
- `setTag:` (page 838)
  Sets the tag of the receiver.

Activating from the Keyboard

- `performClick:` (page 824)
  Simulates a single mouse click on the receiver.
- `refusesFirstResponder` (page 825)
  Returns a Boolean value indicating whether the receiver refuses the first responder role.

- `setRefusesFirstResponder:` (page 837)
  Sets whether the receiver refuses first responder role.

Tracking the Mouse

- `mouseDown:` (page 823)
  Informs the receiver that the user has pressed the left mouse button.
- `ignoresMultiClick` (page 820)
  Returns a Boolean value indicating whether the receiver ignores multiple clicks made in rapid succession.
- `setIgnoresMultiClick:` (page 834)
  Sets whether the receiver ignores multiple clicks made in rapid succession.

Control Editing Notifications

- `controlTextDidBeginEditing:` (page 846) *delegate method*
  Sent when a control with editable text begins an editing session.
- `controlTextDidChange:` (page 846) *delegate method*
  Sent when the text in the receiving control changes.
- `controlTextDidEndEditing:` (page 847) *delegate method*
  Sent when a control with editable text ends an editing session.

Class Methods

cellClass

Returns the type of cell used by the receiver.

```
+ (Class)cellClass
```

Return Value
The class of the cell used to manage the receiver's contents, or `nil` if no cell class has been set for the receiver or its superclasses (up to NSControl).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cell` (page 816)
- `setCell:` (page 830)
- `setCellClass:` (page 813)

Related Sample Code
ClockControl
TrackBall

Declared In
NSControl.h

setCellClass:

Sets the type of cell used by the receiver.

```
+ (void)setCellClass:(Class)class
```

Parameters

class

The class of the cell to use with this control.

Discussion
If you have a custom cell subclass that you would like to substitute for the class of a cell object in a nib file, you should set the cell class in the awakeFromNib (page 3731) method (NSNibAwaking protocol). You cannot change the class programmatically after the cell object has been unarchived from the nib and instantiated, which occurs immediately after awakeFromNib (page 3731) returns. If you are going to be using your custom cell frequently, consider creating your own Interface Builder palette containing the cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- cell (page 816)
- setCell: (page 830)
+ cellClass (page 812)

Declared In
NSControl.h

Instance Methods

abortEditing

Terminates the current editing operation and discards any edited text.

```
- (BOOL)abortEditing
```

Return Value
YES if there was a field editor associated with the control; otherwise, NO.

Discussion
If there was a field editor, this method removes the field editor's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- currentEditor (page 817)
- validateEditing (page 845)

Related Sample Code
QTMetadataEditor

Declared In
`NSControl.h`

action

Returns the default action-message selector associated with the control.

```
- (SEL)action
```

Return Value
The selector associated with the receiver's cell.

Discussion
The `NSControl` implementation of this method returns the action message selector of the receiver's cell. Controls that support multiple cells (such as `NSMatrix` and `NSForm`) must override this method to return the appropriate action-message selector.

If you want the action-message selector for a control that has multiple cells, it is better to use the get the selector directly from the cell's own `action` method, as shown in the following example:

```
SEL someAction = [[theControl selectedCell] action];
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAction:` (page 828)
- `setTarget:` (page 838)
- `target` (page 844)

Related Sample Code
SimpleToolbar

Declared In
`NSControl.h`

alignment

Returns the alignment mode of the text in the receiver's cell.

```
- (NSTextAlignment)alignment
```

Return Value
One of the following constants: `NSLeftTextAlignment`, `NSRightTextAlignment`,`NSCenterTextAlignment`, `NSJustifiedTextAlignment`, or `NSNaturalTextAlignment`. The default value is `NSNaturalTextAlignment`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlignment:` (page 828)

Declared In
`NSControl.h`

attributedStringValue

Returns the value of the receiver's cell as an attributed string.

```
- (NSAttributedString *)attributedStringValue
```

Return Value
The value of the cell interpreted as an attributed string, or an empty attributed string if the receiver has no cell.

Discussion
If the control contains many cells (for example, `NSMatrix`), then the value of the currently selected cell is returned. If the control is in the process of editing the affected cell, then it invokes the `validateEditing` (page 845) method before extracting and returning the value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAttributedStringValue:` (page 829)

Declared In
`NSControl.h`

baseWritingDirection

Returns the initial writing direction used to determine the actual writing direction for text.

```
- (NSWritingDirection)baseWritingDirection
```

Return Value
One of the following values: `NSWritingDirectionNatural`, `NSWritingDirectionLeftToRight`, or `NSWritingDirectionRightToLeft`. The default value is `NSWritingDirectionNatural`.

Discussion
The Text system uses this value as a hint for calculating the actual direction for displaying Unicode characters. You should not need to call this method directly.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setBaseWritingDirection:` (page 829)

Declared In
`NSControl.h`

calcSize

Recomputes any internal sizing information for the receiver, if necessary.

```
- (void)calcSize
```

Discussion
This method uses the `calcDrawInfo:` (page 548) method of its cell to perform the calculations. Most controls maintain a flag that informs them if any of their cells have been modified in such a way that the location or size of the cell should be recomputed. If such a modification happens, this method is automatically invoked before the control is displayed. You should never need to invoke it yourself.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- sizeToFit` (page 839)

Declared In
`NSControl.h`

cell

Returns the receiver's cell object.

```
- (id)cell
```

Return Value
The cell object of the receiver.

Discussion
For controls with multiple cells (such as `NSMatrix` or `NSForm`), use the `selectedCell` (page 825) method or a similar method to retrieve a specific cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ cellClass` (page 812)
`+ setCellClass:` (page 813)
`- setCell:` (page 830)

Related Sample Code
ButtonMadness
ClockControl
DatePicker
FunHouse
Quartz Composer QCTV

Declared In
`NSControl.h`

currentEditor

Returns the current field editor for the control.

```
- (NSText *)currentEditor
```

Return Value
The field editor for the current control, or `nil` if the receiver does not have a field editor.

Discussion
When the receiver is a control displaying editable text (for example, a text field) and it is the first responder, it has a field editor, which is returned by this method. The field editor is a single `NSTextView` object that is shared among all the controls in a window for light text-editing needs. It is automatically instantiated when needed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `abortEditing` (page 813)
- `validateEditing` (page 845)

Declared In
`NSControl.h`

doubleValue

Returns the value of the receiver's cell as a double-precision floating-point number.

```
- (double)doubleValue
```

Return Value
The value of the cell interpreted as a double-precision floating-point number.

Discussion
If the control contains many cells (for example, `NSMatrix`), then the value of the currently selected cell is returned. If the control is in the process of editing the affected cell, then it invokes the `validateEditing` (page 845) method before extracting and returning the value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `floatValue` (page 819)
- `intValue` (page 821)
- `integerValue` (page 821)
- `objectValue` (page 824)
- `stringValue` (page 840)
- `setDoubleValue:` (page 831)

Related Sample Code
CarbonCocoaTempConverter
CocoaSpeechSynthesisExample

FunHouse

Declared In
NSControl.h

drawCell:

Draws the specified cell, as long as it belongs to the receiver.

```
- (void)drawCell:(NSCell *)aCell
```

Parameters

aCell

The cell to draw. If the cell does not belong to the receiver, this method does nothing.

Discussion
This method is provided primarily to support a consistent set of methods between NSControl objects with single and multiple cells, because a control with multiple cells needs to be able to draw individual cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectCell: (page 825)
- updateCell: (page 844)
- updateCellInside: (page 845)

Declared In
NSControl.h

drawCellInside:

Draws the inside of the receiver's cell (the area within the bezel or border)

```
- (void)drawCellInside:(NSCell *)aCell
```

Parameters

aCell

The cell to draw. If the cell does not belong to the receiver, this method does nothing.

Discussion
If the receiver is transparent, the method causes the superview to draw itself. This method invokes the drawInteriorWithFrame:inView: (page 553) method of NSCell. This method has no effect on controls (such as NSMatrix and NSForm) that have multiple cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectCell: (page 825)
- updateCell: (page 844)
- updateCellInside: (page 845)

Declared In
NSControl.h

floatValue

Returns the value of the receiver's cell as a single-precision floating-point number.

```
- (float)floatValue
```

Return Value
The value of the cell interpreted as a single-precision floating-point number.

Discussion
If the control contains many cells (for example, NSMatrix), then the value of the currently selected cell is returned. If the control is in the process of editing the affected cell, then it invokes the validateEditing (page 845) method before extracting and returning the value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- doubleValue (page 817)
- intValue (page 821)
- integerValue (page 821)
- objectValue (page 824)
- stringValue (page 840)
- setFloatValue: (page 832)

Related Sample Code
CoreImageGLTextureFBO
Fireworks
NineSlice
OpenALExample
WhackedTV

Declared In
NSControl.h

font

Returns the font used to draw text in the receiver's cell.

```
- (NSFont *)font
```

Return Value
The font object used for drawing text.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFont:` (page 833)

Declared In
`NSControl.h`

formatter

Returns the receiver's formatter.

```
- (id)formatter
```

Return Value
The formatter object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFormatter:` (page 834)

Related Sample Code
TrackBall

Declared In
`NSControl.h`

ignoresMultiClick

Returns a Boolean value indicating whether the receiver ignores multiple clicks made in rapid succession.

```
- (BOOL)ignoresMultiClick
```

Return Value
`YES` if the receiver ignores multiple clicks; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setIgnoresMultiClick:` (page 834)

Declared In
`NSControl.h`

initWithFrame:

Returns an NSControl object initialized with the specified frame rectangle.

```
- (id)initWithFrame:(NSRect)frameRect
```

Parameters

frameRect

The rectangle of the control, specified in points in the coordinate space of the enclosing view.

Return Value

An initialized control object, or `nil` if the object could not be initialized.

Discussion

If a cell has been specified for controls of this type, this method also creates an instance of the cell. Because `NSControl` is an abstract class, invocations of this method should appear only in the designated initializers of subclasses; that is, there should always be a more specific designated initializer for the subclass, as this method is the designated initializer for `NSControl`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSControl.h`

integerValue

Returns the value of the receiver's cell as an `NSInteger` value.

```
- (NSInteger)integerValue
```

Return Value

The value of the cell interpreted as an `NSInteger` value.

Discussion

If the control contains many cells (for example, `NSMatrix`), then the value of the currently selected cell is returned. If the control is in the process of editing the affected cell, then it invokes the `validateEditing` (page 845) method before extracting and returning the value.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `doubleValue` (page 817)
- `floatValue` (page 819)
- `intValue` (page 821)
- `objectValue` (page 824)
- `stringValue` (page 840)
- `setIntegerValue:` (page 834)

Declared In

`NSControl.h`

intValue

Returns the value of the receiver's cell as an integer.

```
- (int)intValue
```

Return Value
The value of the cell interpreted as an integer.

Discussion
If the control contains many cells (for example, `NSMatrix`), then the value of the currently selected cell is returned. If the control is in the process of editing the affected cell, then it invokes the `validateEditing` (page 845) method before extracting and returning the value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `doubleValue` (page 817)
- `floatValue` (page 819)
- `integerValue` (page 821)
- `objectValue` (page 824)
- `stringValue` (page 840)
- `setIntValue:` (page 835)

Related Sample Code
CocoaSpeechSynthesisExample
EnhancedAudioBurn
EnhancedDataBurn
GLUT
QTKitTimeCode

Declared In
`NSControl.h`

isContinuous

Returns a Boolean value indicating whether the receiver's cell sends its action message continuously to its target during mouse tracking.

```
- (BOOL)isContinuous
```

Return Value
`YES` if the action message is sent continuously; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setContinuous:` (page 830)

Related Sample Code
Cropped Image

Declared In
`NSControl.h`

isEnabled

Returns whether the receiver reacts to mouse events.

```
- (BOOL)isEnabled
```

Return Value
`YES` if the receiver responds to mouse events; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setEnabled:` (page 831)

Related Sample Code
EnhancedDataBurn
TrackBall

Declared In
`NSControl.h`

mouseDown:

Informs the receiver that the user has pressed the left mouse button.

```
- (void)mouseDown:(NSEvent *)theEvent
```

Parameters
theEvent
The event resulting from the user action.

Discussion
Invoked when the mouse button is pressed while the cursor is within the bounds of the receiver, generating *theEvent*. This method highlights the receiver's cell and sends it a `trackMouse:inRect:ofView:untilMouseUp:` (page 610) message. Whenever the cell finishes tracking the mouse (for example, because the cursor has left the cell's bounds), the cell is unhighlighted. If the mouse button is still down and the cursor reenters the bounds, the cell is again highlighted and a new `trackMouse:inRect:ofView:untilMouseUp:` (page 610) message is sent. This behavior repeats until the mouse button goes up. If it goes up with the cursor in the control, the state of the control is changed, and the action message is sent to the target. If the mouse button goes up when the cursor is outside the control, no action message is sent.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- ignoresMultiClick` (page 820)
`- trackMouse:inRect:ofView:untilMouseUp:` (page 610) (`NSCell`)

Declared In
`NSControl.h`

objectValue

Returns the value of the receiver's cell as an Objective-C object.

```
- (id)objectValue
```

Return Value
The value of the cell interpreted as an Objective-C object.

Discussion
If the control contains many cells (for example, `NSMatrix`), then the value of the currently selected cell is returned. If the control is in the process of editing the affected cell, then it invokes the `validateEditing` (page 845) method before extracting and returning the value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `doubleValue` (page 817)
- `floatValue` (page 819)
- `intValue` (page 821)
- `stringValue` (page 840)
- `setObjectValue:` (page 836)

Related Sample Code
FunHouse
PhotoSearch
PredicateEditorSample
TrackBall

Declared In
`NSControl.h`

performClick:

Simulates a single mouse click on the receiver.

```
- (void)performClick:(id)sender
```

Parameters
sender
The object requesting the action. This parameter is ignored.

Discussion
This method calls the `performClick:` (page 573) method of the receiver's cell with the sender being the control itself. This method raises an exception if the action message cannot be successfully sent.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSControl.h`

refusesFirstResponder

Returns a Boolean value indicating whether the receiver refuses the first responder role.

```
- (BOOL)refusesFirstResponder
```

Return Value
`YES` if the receiver refuses the first responder role; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setRefusesFirstResponder:` (page 837)

Declared In
`NSControl.h`

selectCell:

Selects the specified cell and redraws the control as needed.

```
- (void)selectCell:(NSCell *)aCell
```

Parameters
aCell
The cell to select. The cell must belong to the receiver.

Discussion
If the cell is already selected (or does not belong to the receiver), this method does nothing. If the cell belongs to the receiver and is not selected, this method changes its state to `NSOnState` and redraws the cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- selectedCell` (page 825)

Declared In
`NSControl.h`

selectedCell

Returns the receiver's selected cell.

```
- (id)selectedCell
```

Return Value
The selected cell object.

Discussion
The default implementation of this method simply returns the control's associated cell (or `nil` if no cell has been set). Subclasses of `NSControl` that manage multiple cells (such as `NSMatrix` and `NSForm`) must override this method to return the cell selected by the user.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cell` (page 816)
- `setCell:` (page 830)

Related Sample Code
Cropped Image
PhotoSearch

Declared In
`NSControl.h`

selectedTag

Returns the tag of the receiver's selected cell.

```
- (NSInteger)selectedTag
```

Return Value
The tag of the selected cell, or `-1` if no cell is selected.

Discussion
When you set the tag of a control with a single cell in Interface Builder, it sets the tags of both the control and the cell with the same value as a convenience.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTag:` (page 838)
- `tag` (page 840)

Related Sample Code
CalendarItems
WhackedTV

Declared In
`NSControl.h`

sendAction:to:

Causes the specified action to be sent the target.

```
- (BOOL)sendAction:(SEL)theAction to:(id)theTarget
```

Parameters
theAction
The selector to invoke on the target. If the selector is `NULL`, no message is sent.

theTarget

The target object to receive the message. If the object is `nil`, the application searches the responder chain for an object capable of handling the message. For more information on dispatching actions, see the class description for `NSActionCell`.

Return Value
`YES` if the message was successfully sent; otherwise, `NO`.

Discussion
This method uses the `sendAction:to:from:` (page 166) method of `NSApplication` to invoke the specified method on an object. The receiver is passed as the parameter to the action message. This method is invoked primarily by the `trackMouse:inRect:ofView:untilMouseUp:` (page 610) method of `NSCell`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `action` (page 814)
- `target` (page 844)

Declared In
`NSControl.h`

sendActionOn:

Sets the conditions on which the receiver sends action messages to its target.

```
- (NSInteger)sendActionOn:(NSInteger)mask
```

Parameters

mask

A bit mask containing the conditions for sending the action. The only conditions that are actually checked are associated with the `NSLeftMouseDownMask`, `NSLeftMouseUpMask`, `NSLeftMouseDraggedMask`, and `NSPeriodicMask` bits.

Return Value
A bit mask containing the previous settings. This bit mask uses the same values as specified in the *mask* parameter.

Discussion
You use this method during mouse tracking when the mouse button changes state, the mouse moves, or if the cell is marked to send its action continuously while tracking. Because of this, the only bits checked in *mask* are `NSLeftMouseDownMask`, `NSLeftMouseUpMask`, `NSLeftMouseDraggedMask`, and `NSPeriodicMask`, which are declared in the `NSEvent` class reference.

The default implementation of this method simply invokes the `sendActionOn:` (page 576) method of its associated cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sendAction:to:` (page 826)
- `sendActionOn:` (page 576) (`NSCell`)

Declared In
NSControl.h

setAction:

Sets the receiver's action method to the specified selector.

```
- (void)setAction:(SEL)aSelector
```

Parameters

aSelector

The new action-message selector to associate with the receiver's cell. Specify NULL to prevent action messages from being sent to the receiver's target.

Discussion
See *Action Messages* for additional information on action messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
- action (page 814)
- setTarget: (page 838)
- target (page 844)

Related Sample Code
FunHouse
PrefsPane
QTAudioContextInsert
QTAudioExtractionPanel
Quartz Composer QCTV

Declared In
NSControl.h

setAlignment:

Sets the alignment of text in the receiver's cell.

```
- (void)setAlignment:(NSTextAlignment)mode
```

Parameters

mode

One of the following constants: NSLeftTextAlignment, NSRightTextAlignment,NSCenterTextAlignment, NSJustifiedTextAlignment, or NSNaturalTextAlignment.

Discussion
If the cell is currently being edited, this method aborts the edits to change the alignment.

Availability
Available in Mac OS X v10.0 and later.

See Also
- alignment (page 814)

Related Sample Code
Quartz Composer QCTV

Declared In
NSControl.h

setAttributedStringValue:

Sets the value of the receiver's cell using an attributed string.

```
- (void)setAttributedStringValue:(NSAttributedString *)object
```

Parameters

object

The value of the cell interpreted as an attributed string.

Discussion
If the cell is being edited, this method aborts all editing before setting the value. If the cell does not inherit from NSActionCell, the method marks the cell's interior as needing to be redisplayed; NSActionCell performs its own updating of cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- attributedStringValue (page 815)

Declared In
NSControl.h

setBaseWritingDirection:

Sets the initial writing direction used to determine the actual writing direction for text .

```
- (void)setBaseWritingDirection:(NSWritingDirection)writingDirection
```

Parameters

writingDirection

One of the following values: NSWritingDirectionNatural, NSWritingDirectionLeftToRight, or NSWritingDirectionRightToLeft.

Discussion
If you know the base writing direction of the text you are rendering, you can use this method to specify that direction to the text system.

Availability
Available in Mac OS X v10.4 and later.

See Also
- baseWritingDirection (page 815)

Declared In
NSControl.h

setCell:

Sets the receiver's cell

```
- (void)setCell:(NSCell *)aCell
```

Parameters

aCell

The new cell for the receiver.

Discussion
Use this method with great care as it can irrevocably damage the affected control; specifically, you should only use this method in initializers for subclasses of `NSControl`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cell` (page 816)
- `selectedCell` (page 825)

Declared In
NSControl.h

setContinuous:

Sets whether the receiver's cell sends its action message continuously to its target during mouse tracking.

```
- (void)setContinuous:(BOOL)flag
```

Parameters

flag

`YES` if the action message should be sent continuously; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isContinuous` (page 822)

Related Sample Code
Cropped Image
SurfaceVertexProgram

Declared In
NSControl.h

setDoubleValue:

Sets the value of the receiver's cell using a double-precision floating-point number.

```
- (void)setDoubleValue:(double)aDouble
```

Parameters

aDouble

The value of the cell interpreted as a double-precision floating-point number.

Discussion

If the cell is being edited, this method aborts all editing before setting the value. If the cell does not inherit from `NSActionCell`, the method marks the cell's interior as needing to be redisplayed; `NSActionCell` performs its own updating of cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `doubleValue` (page 817)
- `setFloatValue:` (page 832)
- `setIntValue:` (page 835)
- `setIntegerValue:` (page 834)
- `setObjectValue:` (page 836)
- `setStringValue:` (page 837)

Related Sample Code

CocoaSpeechSynthesisExample
FunHouse
QTKitMovieShuffler
QTMetadataEditor
Quartz 2D Shadings

Declared In

`NSControl.h`

setEnabled:

Sets whether the receiver (and its cell) reacts to mouse events.

```
- (void)setEnabled:(BOOL)flag
```

Parameters

flag

`YES` if you want the receiver to react to mouse events; otherwise, `NO`.

Discussion

If *flag* is `NO`, any editing is aborted. This method redraws the entire control if it is marked as needing redisplay. Subclasses may want to override this method to redraw only a portion of the control when the enabled state changes; `NSButton` and `NSSlider` do this.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `isEnabled` (page 823)

Related Sample Code
CocoaSpeechSynthesisExample
ColorMatching
GLUT
OpenALExample
WhackedTV

Declared In
`NSControl.h`

setFloatingPointFormat:left:right:

Sets the auto-ranging and floating point number format of the receiver's cell. (Deprecated in Mac OS X v10.0.)

```
- (void)setFloatingPointFormat:(BOOL)autoRange left:(NSUInteger)leftDigits
    right:(NSUInteger)rightDigits
```

Parameters

autoRange
`YES` to enable auto-ranging; otherwise, `NO`.

leftDigits
The number of digits to display to the left of the decimal point.

rightDigits
The number of digits to display to the right of the decimal point.

Discussion
For more information about auto-ranging and how it works, see the description of this method in the `NSCell` class specification. If the cell is being edited, the current edits are discarded and the cell's interior is redrawn.

Note: This method is being deprecated in favor of a new class of formatter objects. For more information, see `NSFormatter`. This documentation is provided only for developers who need to modify older applications.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.0.

See Also
- `setFloatingPointFormat:left:right:` (page 586) (`NSCell`)

Declared In
`NSControl.h`

setFloatValue:

Sets the value of the receiver's cell using a single-precision floating-point number.

```
- (void)setFloatValue:(float)aFloat
```

Parameters

aFloat

The value of the cell interpreted as a single-precision floating-point number.

Discussion

If the cell is being edited, this method aborts all editing before setting the value. If the cell does not inherit from `NSActionCell`, the method marks the cell's interior as needing to be redisplayed; `NSActionCell` performs its own updating of cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `floatValue` (page 819)
- `setDoubleValue:` (page 831)
- `setIntValue:` (page 835)
- `setIntegerValue:` (page 834)
- `setObjectValue:` (page 836)
- `setStringValue:` (page 837)

Related Sample Code

FilterDemo
From A View to A Movie
From A View to A Picture
NineSlice
OpenALExample

Declared In

`NSControl.h`

setFont:

Sets the font used to draw text in the receiver's cell.

```
- (void)setFont:(NSFont *)fontObject
```

Parameters

fontObject

The font object to use.

Discussion

If the cell is being edited, the text in the cell is redrawn in the new font, and the cell's editor (the `NSText` object used globally for editing) is updated with the new font object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `font` (page 819)

Related Sample Code

ObjectPath

Declared In
NSControl.h

setFormatter:

Sets the receiver's formatter

```
- (void)setFormatter:(NSFormatter *)newFormatter
```

Parameters

newFormatter
The new formatter object to use with the control.

Availability
Available in Mac OS X v10.0 and later.

See Also
- formatter (page 820)

Declared In
NSControl.h

setIgnoresMultiClick:

Sets whether the receiver ignores multiple clicks made in rapid succession.

```
- (void)setIgnoresMultiClick:(BOOL)flag
```

Parameters

flag
YES if the receiver should ignore multiple clicks; otherwise, NO.

Discussion
By default, controls treat double clicks as two distinct clicks, triple clicks as three distinct clicks, and so on. However, if you pass YES to this method, additional clicks (within a predetermined interval after the first) occurring after the first click are not processed by the receiver, and are instead passed on to super.

Availability
Available in Mac OS X v10.0 and later.

See Also
- ignoresMultiClick (page 820)

Declared In
NSControl.h

setIntegerValue:

Sets the value of the receiver's cell using an NSInteger value.

```
- (void)setIntegerValue:(NSInteger)anInteger
```

Parameters

anInteger

The value of the cell interpreted as an `NSInteger` value.

Discussion

If the cell is being edited, this method aborts all editing before setting the value. If the cell does not inherit from `NSActionCell`, the method marks the cell's interior as needing to be redisplayed; `NSActionCell` performs its own updating of cells.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `integerValue` (page 821)
- `setDoubleValue:` (page 831)
- `setFloatValue:` (page 832)
- `setIntValue:` (page 835)
- `setObjectValue:` (page 836)
- `setStringValue:` (page 837)

Declared In

`NSControl.h`

setIntValue:

Sets the value of the receiver's cell using an integer.

```
- (void)setIntValue:(int)anInt
```

Parameters

anInt

The value of the cell interpreted as an integer.

Discussion

If the cell is being edited, this method aborts all editing before setting the value. If the cell does not inherit from `NSActionCell`, the method marks the cell's interior as needing to be redisplayed; `NSActionCell` performs its own updating of cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `intValue` (page 821)
- `setDoubleValue:` (page 831)
- `setIntegerValue:` (page 834)
- `setFloatValue:` (page 832)
- `setObjectValue:` (page 836)
- `setStringValue:` (page 837)

Related Sample Code

AESendThreadSafe

Fireworks

From A View to A Movie
From A View to A Picture
ObjectPath

Declared In
NSControl.h

setNeedsDisplay

Marks the receiver as needing redisplay (assuming automatic display is enabled).

```
- (void)setNeedsDisplay
```

Discussion
This method also recalculates the dimensions of the control as needed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setNeedsDisplay: (page 3225) (NSView)

Declared In
NSControl.h

setObjectValue:

Sets the value of the receiver's cell using an Objective-C object.

```
- (void)setObjectValue:(id < NSCopying >)object
```

Parameters
object
The value of the cell interpreted as an Objective-C object.

Discussion
If the cell is being edited, this method aborts all editing before setting the value. If the cell does not inherit from NSActionCell, the method marks the cell's interior as needing to be redisplayed; NSActionCell performs its own updating of cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- objectValue (page 824)
- setDoubleValue: (page 831)
- setFloatValue: (page 832)
- setIntValue: (page 835)
- setStringValue: (page 837)

Related Sample Code
EnhancedAudioBurn

EnhancedDataBurn
NewsReader
Quartz Composer QCTV

Declared In
NSControl.h

setRefusesFirstResponder:

Sets whether the receiver refuses first responder role.

```
- (void)setRefusesFirstResponder:(BOOL)flag
```

Parameters

flag

YES if the receiver should refuse the first responder role; otherwise, NO.

Discussion
By default, the user can advance the focus of keyboard events between controls by pressing the Tab key; when this focus—or first responder status—is indicated for a control (by the insertion point or, for nontext controls, a faint rectangle), the user can activate the control by pressing the Space bar.

Availability
Available in Mac OS X v10.0 and later.

See Also
- refusesFirstResponder (page 825)
- objectValue (page 824)
- setDoubleValue: (page 831)
- setFloatValue: (page 832)

Declared In
NSControl.h

setStringValue:

Sets the value of the receiver's cell using an NSString object.

```
- (void)setStringValue:(NSString *)aString
```

Parameters

aString

The value of the cell interpreted as an NSString object.

Discussion
If the cell is being edited, this method aborts all editing before setting the value. If the cell does not inherit from NSActionCell, the method marks the cell's interior as needing to be redisplayed; NSActionCell performs its own updating of cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDoubleValue:` (page 831)
- `setFloatValue:` (page 832)
- `setIntValue:` (page 835)
- `setObjectValue:` (page 836)
- `stringValue` (page 840)

Related Sample Code
EnhancedAudioBurn
FunHouse
GLUT
OpenALExample
PDFKitLinker2

Declared In
`NSControl.h`

setTag:

Sets the tag of the receiver.

```
- (void)setTag:(NSInteger)anInt
```

Parameters
anInt
The new tag.

Discussion
This method does not affect the tag of the receiver's cell. Tags allow you to identify particular cells. Tag values are not used internally; they are only changed by external invocations of `setTag:` (page 838). You typically set tag values in Interface Builder and use them at runtime in your application. When you set the tag of a control with a single cell in Interface Builder, it sets the tags of both the control and the cell to the same value as a convenience.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `tag` (page 840)
- `selectedTag` (page 826)

Related Sample Code
FunHouse
Quartz Composer QCTV

Declared In
`NSControl.h`

setTarget:

Sets the target object to receive action messages from the receiver's cell.

```
- (void)setTarget:(id)anObject
```

Parameters

anObject

The new target object to associate with the receiver's cell, or `nil` to remove the current target.

Discussion

If *anObject* is `nil` but the control still has a valid action message assigned, the application follows the responder chain looking for an object that can respond to the message. See the description of the `NSActionCell` class for details.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `action` (page 814)
- `setAction:` (page 828)
- `target` (page 844)
- `setTarget:` (page 598) (NSCell)

Related Sample Code

FunHouse

PrefsPane

Quartz Composer QCTV

ScriptingBridgeFinder

SimpleCocoaApp

Declared In

`NSControl.h`

sizeToFit

Resizes the receiver's frame so that it is the minimum size needed to contain its cell.

```
- (void)sizeToFit
```

Discussion

If you want a multiple-cell custom subclass of `NSControl` to size itself to fit its cells, you must override this method. This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with either the`display` (page 3163) or `setNeedsDisplay` (page 836) method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `calcSize` (page 816)

Related Sample Code

Quartz Composer QCTV

Declared In

`NSControl.h`

stringValue

Returns the value of the receiver's cell as an `NSString` object.

```
- (NSString *)stringValue
```

Return Value
The value of the cell interpreted as an `NSString` object.

Discussion
If the control contains many cells (for example, `NSMatrix`), then the value of the currently selected cell is returned. If the control is in the process of editing the affected cell, then it invokes the `validateEditing` (page 845) method before extracting and returning the value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `doubleValue` (page 817)
- `floatValue` (page 819)
- `intValue` (page 821)
- `objectValue` (page 824)
- `setStringValue:` (page 837)

Related Sample Code
CoreRecipes
EnhancedAudioBurn
FinalCutPro_AppleEvents
MovieAssembler
WhackedTV

Declared In
`NSControl.h`

tag

Returns the tag identifying the receiver (not the tag of the receiver's cell).

```
- (NSInteger)tag
```

Return Value
The tag of this control object.

Discussion
Tags allow you to identify particular controls. Tag values are not used internally; they are only changed by external invocations of `setTag:`. You typically set tag values in Interface Builder and use them at runtime in your application. When you set the tag of a control with a single cell in Interface Builder, it sets the tags of both the control and the cell to the same value as a convenience.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTag:` (page 838)
- `selectedTag` (page 826)

Related Sample Code
Cropped Image
EnhancedDataBurn
OpenALExample
Quartz Composer QCTV
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSControl.h`

takeDoubleValueFrom:

Sets the value of the receiver's cell to a double-precision floating-point value obtained from the specified object.

```
- (void)takeDoubleValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the `doubleValue` (page 817) message.

Discussion
You can use this method to link action messages between controls. It permits one control or cell (*sender*) to affect the value of another control (the receiver) by invoking this method in an action message to the receiver. For example, a text field can be made the target of a slider. Whenever the slider is moved, it sends this message to the text field. The text field then obtains the slider's value, turns it into a text string, and displays it.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSControl.h`

takeFloatValueFrom:

Sets the value of the receiver's cell to a single-precision floating-point value obtained from the specified object.

```
- (void)takeFloatValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the `floatValue` (page 819) message.

Discussion
You can use this method to link action messages between controls. It permits one control or cell (*sender*) to affect the value of another control (the receiver) by invoking this method in an action message to the receiver. For example, a text field can be made the target of a slider. Whenever the slider is moved, it sends this message to the text field. The text field then obtains the slider's value, turns it into a text string, and displays it.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSControl.h

takeIntegerValueFrom:

Sets the value of the receiver's cell to an NSInteger value obtained from the specified object.

```
- (void)takeIntegerValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the integerValue (page 821) message.

Discussion
You can use this method to link action messages between controls. It permits one control or cell (*sender*) to affect the value of another control (the receiver) by invoking this method in an action message to the receiver. For example, a text field can be made the target of a slider. Whenever the slider is moved, it sends this message to the text field. The text field then obtains the slider's value, turns it into a text string, and displays it.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSControl.h

takeIntValueFrom:

Sets the value of the receiver's cell to an integer value obtained from the specified object.

```
- (void)takeIntValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the intValue (page 821) message.

Discussion
You can use this method to link action messages between controls. It permits one control or cell (*sender*) to affect the value of another control (the receiver) by invoking this method in an action message to the receiver. For example, a text field can be made the target of a slider. Whenever the slider is moved, it sends this message to the text field. The text field then obtains the slider's value, turns it into a text string, and displays it.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSControl.h

takeObjectValueFrom:

Sets the value of the receiver's cell to the object value obtained from the specified object.

```
- (void)takeObjectValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the `objectValue` (page 824) message.

Discussion
You can use this method to link action messages between controls. It permits one control or cell (*sender*) to affect the value of another control (the receiver) by invoking this method in an action message to the receiver. For example, a text field can be made the target of a slider. Whenever the slider is moved, it sends this message to the text field. The text field then obtains the slider's value, turns it into a text string, and displays it.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSControl.h

takeStringValueFrom:

Sets the value of the receiver's cell to the string value obtained from the specified object.

```
- (void)takeStringValueFrom:(id)sender
```

Parameters

sender

The object from which to take the value. This object must respond to the `stringValue` (page 840) message.

Discussion
You can use this method to link action messages between controls. It permits one control or cell (*sender*) to affect the value of another control (the receiver) by invoking this method in an action message to the receiver. For example, a text field can be made the target of a slider. Whenever the slider is moved, it sends this message to the text field. The text field then obtains the slider's value, turns it into a text string, and displays it.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSControl.h

target

Returns the target object of the receiver's cell.

```
- (id)target
```

Return Value
The target object that receives action messages from the cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- action (page 814)
- setAction: (page 828)
- setTarget: (page 838)

Related Sample Code
RadiantColorPicker
SimpleToolbar
UIElementInspector

Declared In
NSControl.h

updateCell:

Marks the specified cell as in need of redrawing.

```
- (void)updateCell:(NSCell *)aCell
```

Parameters
aCell
The cell to redraw.

Discussion
If the cell currently has the focus, this method updates the cell's focus ring; otherwise, the entire cell is marked as needing redisplay. The cell is redrawn during the next update cycle.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSControl.h

updateCellInside:

Marks the inside of the specified cell as in need of redrawing.

```
- (void)updateCellInside:(NSCell *)aCell
```

Parameters

aCell

The cell to redraw.

Availability
Available in Mac OS X v10.0 and later.

See Also
- updateCell: (page 844)

Declared In
NSControl.h

validateEditing

Validates changes to any user-typed text.

```
- (void)validateEditing
```

Discussion
Validation sets the object value of the cell to the current contents of the cell's editor (the NSText object used for editing), storing it as a simple NSString or an attributed string object based on the attributes of the editor.

Availability
Available in Mac OS X v10.0 and later.

See Also
- abortEditing (page 813)
- currentEditor (page 817)

Declared In
NSControl.h

Delegate Methods

controlTextDidBeginEditing:

Sent when a control with editable text begins an editing session.

```
- (void)controlTextDidBeginEditing:(NSNotification *)aNotification
```

Parameters

aNotification

The notification object. The name of the notification is always `NSControlTextDidBeginEditingNotification` (page 847).

Discussion

This method is invoked when the user begins editing text in a control such as a text field or a form field. The control posts a `NSControlTextDidBeginEditingNotification` (page 847) notification, and if the control's delegate implements this method, it is automatically registered to receive the notification. Use the key `@"NSFieldEditor"` to obtain the field editor from the `userInfo` dictionary of the notification object.

See `controlTextDidEndEditing:` (page 847) for an explanation of why you may not always get one invocation of `controlTextDidBeginEditing:` (page 846) for each invocation of `controlTextDidEndEditing:` (page 847).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSControl.h`

controlTextDidChange:

Sent when the text in the receiving control changes.

```
- (void)controlTextDidChange:(NSNotification *)aNotification
```

Parameters

aNotification

The notification object. The name of the notification is always `NSControlTextDidChangeNotification` (page 848).

Discussion

This method is invoked when text in a control such as a text field or form changes. The control posts a `NSControlTextDidChangeNotification` (page 848) notification, and if the control's delegate implements this method, it is automatically registered to receive the notification. Use the key `@"NSFieldEditor"` to obtain the field editor from the `userInfo` dictionary of the notification object.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSControl.h`

controlTextDidEndEditing:

Sent when a control with editable text ends an editing session.

```
- (void)controlTextDidEndEditing:(NSNotification *)aNotification
```

Parameters

aNotification

The notification object. The name of the notification is always `NSControlTextDidEndEditingNotification` (page 848).

Discussion

This method is invoked when the user stops editing text in a control such as a text field or form. The control posts a `NSControlTextDidEndEditingNotification` (page 848) notification, and if the control's delegate implements this method, it is automatically registered to receive the notification. Use the key `@"NSFieldEditor"` to obtain the field editor from the `userInfo` dictionary of the notification object.

Warning: In some cases, such as when editing within an instance of `NSOutlineView`, this method may be invoked without a previous invocation of `controlTextDidBeginEditing:` (page 846). You will only get the `controlTextDidBeginEditing:` notification if the user actually types something, but you can get the `controlTextDidEndEditing:` notification if the user just double-clicks the field and then clicks outside the field, without typing.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSControl.h`

Notifications

An `NSControl` object posts the following notifications to interested observers and its delegate. Note that although the `NSControl` class defines delegate methods, it does not itself have a delegate. Any subclass that uses these methods must have a delegate and the methods to get and set it.

NSControlTextDidBeginEditingNotification

Sent when a control with editable cells begins an edit session.

The field editor of the edited cell originally sends an `NSTextDidBeginEditingNotification` (page 2752) to the control, which passes it on in this form to its delegate. The notification object is the `NSControl` object posting the notification. The *userInfo* dictionary contains the following information:

Key	Value
`@"NSFieldEditor"`	The edited cell's field editor

See the `controlTextDidEndEditing:` (page 847) method for details.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSControl.h`

NSControlTextDidChangeNotification

Sent when the text in the receiving control changes.

The field editor of the edited cell originally sends an `NSTextDidChangeNotification` (page 2752) to the control, which passes it on in this form to its delegate. The notification object is the `NSControl` object posting the notification. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSFieldEditor"`	The edited cell's field editor

See the `controlTextDidChange:` (page 846) method for details.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSControl.h`

NSControlTextDidEndEditingNotification

Sent when a control with editable cells ends an editing session.

The field editor of the edited cell originally sends an `NSControlTextDidEndEditingNotification` (page 848) to the control, which passes it on in this form to its delegate. The notification object is the `NSControl` object posting the notification. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSFieldEditor"`	The edited cell's field editor

See the `controlTextDidEndEditing:` (page 847) method for details.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSControl.h`

NSController Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSController.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Cocoa Bindings Programming Topics
Related sample code	CIRAWFilterSample

Overview

NSController is an abstract class that implements the NSEditor and NSEditorRegistration informal protocols required for controller classes.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Managing Editing

`- objectDidBeginEditing:` (page 852)
Invoked to inform the receiver that *`editor`* has uncommitted changes that can affect the receiver.

`- objectDidEndEditing:` (page 852)
Invoked to inform the receiver that *`editor`* has committed or discarded its changes.

- `commitEditing` (page 850)
  Causes the receiver to attempt to commit any pending edits, returning `YES` if successful or no edits were pending.
- `commitEditingWithDelegate:didCommitSelector:contextInfo:` (page 850)
  Attempts to commit any pending changes in known editors of the receiver.
- `discardEditing` (page 851)
  Discards any pending changes by registered editors.
- `isEditing` (page 852)
  Returns `YES` if there are any editors currently registered with the receiver, `NO` otherwise.

Instance Methods

commitEditing

Causes the receiver to attempt to commit any pending edits, returning `YES` if successful or no edits were pending.

```
- (BOOL)commitEditing
```

Discussion
The receiver invokes `commitEditing` (page 3678) on any current editors, returning their response. A commit is denied if the receiver fails to apply the changes to the model object, perhaps due to a validation error.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `discardEditing` (page 851)

Declared In
`NSController.h`

commitEditingWithDelegate:didCommitSelector:contextInfo:

Attempts to commit any pending changes in known editors of the receiver.

```
-(void)commitEditingWithDelegate:(id)delegate
    didCommitSelector:(SEL)didCommitSelector contextInfo:(void *)contextInfo
```

Parameters

delegate
An object that can serve as the receiver's delegate. It should implement the method specified by *didCommitSelector*.

didCommitSelector
A selector that is invoked on delegate. The method specified by the selector must have the same signature as the following method:

```
- (void)editor:(id)editor didCommit:(BOOL)didCommit contextInfo:(void
*)contextInfo
```

contextInfo

Contextual information that is sent as the `contextInfo` argument to delegate when *didCommitSelector* is invoked.

Discussion
Provides support for the `NSEditor` informal protocol. This method attempts to commit pending changes in known editors. Known editors are either instances of a subclass of `NSController` or (more rarely) user interface controls that may contain pending edits—such as text fields—that registered with the context using `objectDidBeginEditing:` and have not yet unregistered using a subsequent invocation of `objectDidEndEditing:`.

The receiver iterates through the array of its known editors and invokes `commitEditing` on each. The receiver then sends the message specified by the *didCommitSelector* selector to the specified delegate.

The `didCommit` argument is the value returned by the editor specified by `editor` from the `commitEditing` message. The `contextInfo` argument is the same value specified as the *contextInfo* parameter—you may use this value however you wish.

If an error occurs while attempting to commit, for example if key-value coding validation fails, your implementation of this method should typically send the view in which editing is being performed a `presentError:modalForWindow:delegate:didRecoverSelector:contextInfo:` message, specifying the view's containing window.

You may find this method useful in some situations (typically if you are using Cocoa Bindings) when you want to ensure that pending changes are applied before a change in user interface state. For example, you may need to ensure that changes pending in a text field are applied before a window is closed. See also `commitEditing` (page 850) which performs a similar function but which allows you to handle any errors directly, although it provides no information beyond simple success/failure.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `commitEditing` (page 850)
- `discardEditing` (page 851)
- `objectDidBeginEditing:` (page 852)
- `objectDidEndEditing:` (page 852)

Declared In
`NSController.h`

discardEditing

Discards any pending changes by registered editors.

```
- (void)discardEditing
```

Discussion
The receiver invokes `discardEditing` (page 3679) on any current editors.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `commitEditing` (page 850)

Declared In
`NSController.h`

isEditing

Returns `YES` if there are any editors currently registered with the receiver, `NO` otherwise.

```
- (BOOL)isEditing
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSController.h`

objectDidBeginEditing:

Invoked to inform the receiver that *editor* has uncommitted changes that can affect the receiver.

```
- (void)objectDidBeginEditing:(id)editor
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `objectDidEndEditing:` (page 852)

Declared In
`NSController.h`

objectDidEndEditing:

Invoked to inform the receiver that *editor* has committed or discarded its changes.

```
- (void)objectDidEndEditing:(id)editor
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `objectDidBeginEditing:` (page 852)

Declared In
`NSController.h`

NSCursor Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSCursor.h
Companion guide	Cursor Management
Related sample code	DragItemAround GLUT Sketch+Accessibility Sketch-112 TextLinks

Overview

Instances of the `NSCursor` class manage the appearance of the cursor.

The following table shows and describes the system cursors, and indicates the class method for obtaining them:

Cursor	Description
	The arrow cursor (`arrowCursor` (page 857))
	The I-beam cursor for indicating insertion points (`IBeamCursor` (page 860))
	The cross-hair cursor (`crosshairCursor` (page 858))
	The closed-hand cursor (`closedHandCursor` (page 857))

Cursor	Description
	The open-hand cursor (`openHandCursor` (page 861))
	The pointing-hand cursor (`pointingHandCursor` (page 862))
	The resize-left cursor (`resizeLeftCursor` (page 863))
	The resize-right cursor (`resizeRightCursor` (page 863))
	The resize-left-and-right cursor (`resizeLeftRightCursor` (page 863))
	The resize-up cursor (`resizeUpCursor` (page 864))
	The resize-down cursor (`resizeDownCursor` (page 862))
	The resize-up-and-down cursor (`resizeUpDownCursor` (page 864))
	The disappearing item cursor (`disappearingItemCursor` (page 859))

In Mac OS X version 10.3 and later, cursor size is no longer limited to 16 by 16 pixels.

Cursor Rectangles

In Cocoa, you can change the currently displayed cursor based on the position of the mouse over one of your views. You might use this technique to provide visual feedback about what actions the user can take with the mouse. For example, you might display one of the resize cursors whenever the mouse moves over a portion of your view that acts as a custom resizing handle. To set this up, you associate a cursor object with one or more cursor rectangles in the view.

Cursor rectangles are a specialized type of tracking rectangles, which are used to monitor the mouse location in a view. Views implement cursor rectangles using tracking rectangles but provide methods for setting and refreshing cursor rectangles that are distinct from the generic tracking rectangle interface. For information on how to set up cursor rectangles, see Mouse-Tracking and Cursor-Update Events.

Tasks

Initializing a New Cursor

- `initWithImage:hotSpot:` (page 867)
  Initializes a cursor with the given image and hot spot.
- `initWithImage:foregroundColorHint:backgroundColorHint:hotSpot:` (page 866)
  Initializes the cursor with the specified image and hot spot.

Setting Cursor Attributes

- `image` (page 866)
  Returns the receiver's image.
- `hotSpot` (page 865)
  Returns the position of the cursor's hot spot.
+ `hide` (page 860)
  Makes the current cursor invisible.
+ `unhide` (page 865)
  Negates an earlier call to `hide` (page 860) by showing the current cursor.
+ `setHiddenUntilMouseMoves:` (page 864)
  Sets whether the cursor is hidden until the mouse moves.

Controlling Which Cursor Is Current

+ `pop` (page 862)
  Pops the current cursor off the top of the stack.
- `pop` (page 869)
  Sends a `pop` (page 862) message to the receiver's class.
- `push` (page 870)
  Puts the receiver on top of the cursor stack and makes it the current cursor.
- `set` (page 870)
  Makes the receiver the current cursor.
- `mouseEntered:` (page 868)
  Automatically sent to the receiver when the cursor enters a cursor rectangle owned by the receiver.
- `setOnMouseEntered:` (page 871)
  Specifies whether the receiver accepts mouseEntered: (page 868) events.
- `isSetOnMouseEntered` (page 867)
  Returns a Boolean value indicating whether the receiver becomes current on receiving a `mouseEntered:` (page 868) message.
- `mouseExited:` (page 869)
  Automatically sent to the receiver when the cursor exits a cursor rectangle owned by the receiver.

- `setOnMouseExited:` (page 871)
  Sets whether the receiver accepts `mouseExited:` (page 869) events.
- `isSetOnMouseExited` (page 868)
  Returns a Boolean value indicating whether the receiver becomes current when it receives a `mouseExited:` (page 869) message.

Retrieving Cursor Instances

+ `currentCursor` (page 858)
  Returns the application's current cursor.
+ `currentSystemCursor` (page 859)
  Returns the current system cursor.
+ `arrowCursor` (page 857)
  Returns the default cursor, the arrow cursor.
+ `contextualMenuCursor` (page 858)
  Returns the contextual menu system cursor.
+ `closedHandCursor` (page 857)
  Returns the closed-hand system cursor.
+ `crosshairCursor` (page 858)
  Returns the cross-hair system cursor.
+ `disappearingItemCursor` (page 859)
  Returns a cursor indicating that the current operation will result in a disappearing item.
+ `dragCopyCursor` (page 859)
  Returns a cursor indicating that the current operation will result in a copy action.
+ `dragLinkCursor` (page 860)
  Returns a cursor indicating that the current operation will result in a link action.
+ `IBeamCursor` (page 860)
  Returns a cursor that looks like a capital I with a tiny crossbeam at its middle.
+ `openHandCursor` (page 861)
  Returns the open-hand system cursor.
+ `operationNotAllowedCursor` (page 861)
  Returns the operation not allowed cursor.
+ `pointingHandCursor` (page 862)
  Returns the pointing-hand system cursor.
+ `resizeDownCursor` (page 862)
  Returns the resize-down system cursor.
+ `resizeLeftCursor` (page 863)
  Returns the resize-left system cursor.
+ `resizeLeftRightCursor` (page 863)
  Returns the resize-left-and-right system cursor.
+ `resizeRightCursor` (page 863)
  Returns the resize-right system cursor.
+ `resizeUpCursor` (page 864)
  Returns the resize-up system cursor.

`+ resizeUpDownCursor` (page 864)
Returns the resize-up-and-down system cursor.

Class Methods

arrowCursor

Returns the default cursor, the arrow cursor.

```
+ (NSCursor *)arrowCursor
```

Return Value
The default cursor, a slanted arrow with its hot spot at the tip. The arrow cursor is the one you're used to seeing over buttons, scrollers, and many other objects in the window system.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ IBeamCursor` (page 860)
`+ currentCursor` (page 858)
`- hotSpot` (page 865)

Related Sample Code
GLUT
PDFKitLinker2
Sketch+Accessibility
Sketch-112
TextLinks

Declared In
`NSCursor.h`

closedHandCursor

Returns the closed-hand system cursor.

```
+ (NSCursor *)closedHandCursor
```

Return Value
The closed-hand cursor.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
DragItemAround
GeekGameBoard

Declared In
NSCursor.h

contextualMenuCursor

Returns the contextual menu system cursor.

```
+ (NSCursor *)contextualMenuCursor
```

Return Value
The contextual menu cursor

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSCursor.h

crosshairCursor

Returns the cross-hair system cursor.

```
+ (NSCursor *)crosshairCursor
```

Return Value
The cross-hair cursor. This cursor is used for situations when precise location is required (where the lines cross is the hot spot).

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
TrackIt

Declared In
NSCursor.h

currentCursor

Returns the application's current cursor.

```
+ (NSCursor *)currentCursor
```

Return Value
The top cursor on the application's cursor stack. This cursor may not be the visible cursor on the screen if a different application is currently active.

Discussion
The method only returns the cursor set by your application using `NSCursor` methods. It does return not cursors set by other applications or cursors set by your application using Carbon API.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `set` (page 870)
- `push` (page 870)
- `pop` (page 869)
- `mouseEntered:` (page 868)
- `mouseExited:` (page 869)

Declared In
`NSCursor.h`

currentSystemCursor

Returns the current system cursor.

```
+ (NSCursor *)currentSystemCursor
```

Return Value
A cursor whose image and hot spot match those of the currently-displayed cursor on the system

Discussion
This method returns the current system cursor regardless of which application set the cursor, and whether Cocoa or Carbon APIs were used to set it.

This method replaces the now deprecated `QDGetCursorData` function.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSCursor.h`

disappearingItemCursor

Returns a cursor indicating that the current operation will result in a disappearing item.

```
+ (NSCursor *)disappearingItemCursor
```

Return Value
The system cursor that indicates that the current operation will result in a disappearing item (for example, when dragging an item from the dock or a toolbar).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSCursor.h`

dragCopyCursor

Returns a cursor indicating that the current operation will result in a copy action.

```
+ (NSCursor *)dragCopyCursor
```

Return Value
The drag copy cursor.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSCursor.h

dragLinkCursor

Returns a cursor indicating that the current operation will result in a link action.

```
+ (NSCursor *)dragLinkCursor
```

Return Value
The drag link cursor.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSCursor.h

hide

Makes the current cursor invisible.

```
+ (void)hide
```

Discussion
If another cursor becomes current, that cursor will be invisible, too. It will remain invisible until you invoke the `unhide` (page 865) method.

`hide` (page 860) overrides `setHiddenUntilMouseMoves:` (page 864).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CoreAnimationKioskStyleMenu
iChatTheater
LightTable

Declared In
NSCursor.h

IBeamCursor

Returns a cursor that looks like a capital I with a tiny crossbeam at its middle.

```
+ (NSCursor *)IBeamCursor
```

Return Value
The I-beam cursor. This is the cursor that you're used to seeing over editable or selectable text. The I-beam cursor's default hot spot is where the crossbeam intersects the I.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ arrowCursor` (page 857)
`+ currentCursor` (page 858)

Related Sample Code
GLUT

Declared In
`NSCursor.h`

openHandCursor

Returns the open-hand system cursor.

```
+ (NSCursor *)openHandCursor
```

Return Value
The open-hand cursor.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
DragItemAround
GeekGameBoard

Declared In
`NSCursor.h`

operationNotAllowedCursor

Returns the operation not allowed cursor.

```
+ (NSCursor *)operationNotAllowedCursor
```

Return Value
The operation not allowed cursor.

Discussion
This cursor indicates that the operation that is being attempted, perhaps dragging to an item that can't accept the drag type, is being denied.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSCursor.h

pointingHandCursor

Returns the pointing-hand system cursor.

```
+ (NSCursor *)pointingHandCursor
```

Return Value
The pointing-hand cursor. The tip of the pointing finger is the hot spot.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
CocoaDVDPlayer

Declared In
NSCursor.h

pop

Pops the current cursor off the top of the stack.

```
+ (void)pop
```

Discussion
The new object on the top of the stack becomes the current cursor. If the current cursor is the only cursor on the stack, this method does nothing.

Availability
Available in Mac OS X v10.0 and later.

See Also
- push (page 870)

Related Sample Code
DragItemAround
GeekGameBoard

Declared In
NSCursor.h

resizeDownCursor

Returns the resize-down system cursor.

```
+ (NSCursor *)resizeDownCursor
```

Return Value
The resize-down cursor. This cursor is used when moving or resizing an object to indicate that the user can move only in the indicated direction.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSCursor.h

resizeLeftCursor

Returns the resize-left system cursor.

```
+ (NSCursor *)resizeLeftCursor
```

Return Value
The resize-left cursor. This cursor is used when moving or resizing an object to indicate that the user can move only in the indicated direction.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSCursor.h

resizeLeftRightCursor

Returns the resize-left-and-right system cursor.

```
+ (NSCursor *)resizeLeftRightCursor
```

Return Value
The resize-left-and-right cursor. This cursor is used when moving or resizing an object and the object can be moved left or right.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSCursor.h

resizeRightCursor

Returns the resize-right system cursor.

```
+ (NSCursor *)resizeRightCursor
```

Return Value
The resize-right cursor. This cursor is used when moving or resizing an object to indicate that the user can move only in the indicated direction.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSCursor.h

resizeUpCursor

Returns the resize-up system cursor.

```
+ (NSCursor *)resizeUpCursor
```

Return Value
The resize-up cursor. This cursor is used when moving or resizing an object to indicate that the user can move only in the indicated direction.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSCursor.h

resizeUpDownCursor

Returns the resize-up-and-down system cursor.

```
+ (NSCursor *)resizeUpDownCursor
```

Return Value
The resize-up-and-down cursor. This cursor is used when moving or resizing an object and the object can be moved up or down.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSCursor.h

setHiddenUntilMouseMoves:

Sets whether the cursor is hidden until the mouse moves.

```
+ (void)setHiddenUntilMouseMoves:(BOOL)flag
```

Parameters
flag

YES to hide the cursor until one of the following occurs:

- The mouse moves.
- You invoke the method again, with *flag* set to NO.

Discussion
Do not try to counter this method by invoking `unhide` (page 865). The results are undefined.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ hide` (page 860)

Related Sample Code
LightTable
PhotoSearch

Declared In
`NSCursor.h`

unhide

Negates an earlier call to `hide` (page 860) by showing the current cursor.

```
+ (void)unhide
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ setHiddenUntilMouseMoves:` (page 864)
`+ hide` (page 860)

Related Sample Code
iChatTheater
LightTable

Declared In
`NSCursor.h`

Instance Methods

hotSpot

Returns the position of the cursor's hot spot.

```
- (NSPoint)hotSpot
```

Return Value
The point describing the position of the hot spot, specified according to the cursor's flipped coordinate system.

Discussion
For a more complete explanation, see the class description.

Note that an `NSCursor` object is immutable: you cannot change its hot spot after it's created. Instead, use `initWithImage:hotSpot:` (page 867) to create a new cursor with the new settings.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- initWithImage:hotSpot:` (page 867)

Declared In
`NSCursor.h`

image

Returns the receiver's image.

```
- (NSImage *)image
```

Return Value
The cursor image or `nil` if none exists

Discussion
Note that an `NSCursor` object is immutable: you cannot change its image after it's created. Instead, use `initWithImage:hotSpot:` (page 867) to create a new cursor with the new settings.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- initWithImage:hotSpot:` (page 867)

Declared In
`NSCursor.h`

initWithImage:foregroundColorHint:backgroundColorHint:hotSpot:

Initializes the cursor with the specified image and hot spot.

```
- (id)initWithImage:(NSImage *)newImage foregroundColorHint:(NSColor *)fg
    backgroundColorHint:(NSColor *)bg hotSpot:(NSPoint)hotSpot
```

Parameters

newImage
The image to assign to the cursor.

fg
The foreground color. This is currently ignored.

bg
The background color. This is currently ignored.

hotSpot
The point to assign as the cursor's hot spot.

Return Value
The initialized cursor object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initWithImage:hotSpot:` (page 867)

Declared In
`NSCursor.h`

initWithImage:hotSpot:

Initializes a cursor with the given image and hot spot.

```
- (id)initWithImage:(NSImage *)newImage hotSpot:(NSPoint)aPoint
```

Parameters

newImage
The image to assign to the cursor.

aPoint
The point to set as the cursor's hot spot.

Return Value
An initialized cursor object.

Discussion
This method is the designated initializer for the class.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hotSpot` (page 865)
- `image` (page 866)
- `initWithImage:foregroundColorHint:backgroundColorHint:hotSpot:` (page 866)

Related Sample Code
GLUT
PDFView
QuickLookSketch
Sketch-112
TextLinks

Declared In
`NSCursor.h`

isSetOnMouseEntered

Returns a Boolean value indicating whether the receiver becomes current on receiving a `mouseEntered:` (page 868) message.

```
- (BOOL)isSetOnMouseEntered
```

Return Value
`YES` if the receiver will become current when it receives a `mouseEntered:` (page 868) message; otherwise, `NO`.

Discussion
To receive such a message, the receiver must first be assigned a cursor rectangle. This assignment can be made using the `NSView` method `addCursorRect:cursor:` (page 3139). For a more complete explanation, see the class description.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setOnMouseEntered:` (page 871)
- `isSetOnMouseExited` (page 868)

Declared In
`NSCursor.h`

isSetOnMouseExited

Returns a Boolean value indicating whether the receiver becomes current when it receives a `mouseExited:` (page 869) message.

```
- (BOOL)isSetOnMouseExited
```

Return Value
`YES` if the receiver becomes current when it receives a `mouseExited:` (page 869) message; otherwise, `NO`.

Discussion
To receive such a message, the receiver must first be assigned a cursor rectangle. This assignment can be made using the `NSView` method `addCursorRect:cursor:` (page 3139). For a more complete explanation, see the class description.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setOnMouseExited:` (page 871)

Declared In
`NSCursor.h`

mouseEntered:

Automatically sent to the receiver when the cursor enters a cursor rectangle owned by the receiver.

```
- (void)mouseEntered:(NSEvent *)anEvent
```

Parameters

anEvent

The event generated when the cursor enters the cursor rectangle.

Discussion

If used after `setOnMouseEntered:` (page 871) has been called with an argument of `YES`, `mouseEntered:` can make the receiver the current cursor.

In your programs, you won't invoke `mouseEntered:` explicitly. It's only included in the class interface so you can override it.

For a more complete explanation, see Mouse-Tracking and Cursor-Update Events and the `NSView` method `addTrackingRect:owner:userData:assumeInside:` (page 3142).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `isSetOnMouseEntered` (page 867)
- `mouseExited:` (page 869)

Declared In

`NSCursor.h`

mouseExited:

Automatically sent to the receiver when the cursor exits a cursor rectangle owned by the receiver.

```
- (void)mouseExited:(NSEvent *)anEvent
```

Parameters

anEvent

The event generated when the cursor exits the cursor rectangle.

Discussion

Like `mouseEntered:` (page 868), this message is part of the class interface only so you can override it.

For a more complete explanation, see Mouse-Tracking and Cursor-Update Events and the NSView method `addTrackingRect:owner:userData:assumeInside:` (page 3142).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setOnMouseExited:` (page 871)
- `isSetOnMouseExited` (page 868)

Declared In

`NSCursor.h`

pop

Sends a `pop` (page 862) message to the receiver's class.

```
- (void)pop
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- push (page 870)
- pop (page 869)

Declared In
NSCursor.h

push

Puts the receiver on top of the cursor stack and makes it the current cursor.

```
- (void)push
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- pop (page 869)
- pop (page 869)

Related Sample Code
DragItemAround
GeekGameBoard

Declared In
NSCursor.h

set

Makes the receiver the current cursor.

```
- (void)set
```

Availability
Available in Mac OS X v10.0 and later.

See Also
+ currentCursor (page 858)

Related Sample Code
CocoaDVDPlayer
PDFKitLinker2
TrackIt

Declared In
NSCursor.h

setOnMouseEntered:

Specifies whether the receiver accepts mouseEntered: (page 868) events.

```
- (void)setOnMouseEntered:(BOOL)flag
```

Parameters

flag

YES if the receiver accepts future mouseEntered: (page 868) event messages; otherwise it ignores them.

Discussion

Accepting mouseEntered: (page 868) event messages allows the cursor to be made the current cursor when the cursor enters a view's cursor rectangle.

Availability

Available in Mac OS X v10.0 and later.

See Also

- mouseEntered: (page 868)

Declared In

NSCursor.h

setOnMouseExited:

Sets whether the receiver accepts mouseExited: (page 869) events.

```
- (void)setOnMouseExited:(BOOL)flag
```

Parameters

flag

YES if the receiver accepts future mouseExited: (page 869) event messages; otherwise it ignores them.

Discussion

Accepting mouseExited: (page 869) event messages allows the cursor to be made the current cursor when the cursor exits a view's cursor rectangle.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSCursor.h

Constants

AppKit Versions for NSCursor Bug Fixes

The version of the AppKit framework containing a specific bug fix.

```
#define NSAppKitVersionNumberWithCursorSizeSupport 682.0
```

Constants

`NSAppKitVersionNumberWithCursorSizeSupport`

The specific version of the AppKit framework that introduced support for cursors larger than 16 x 16 pixels in size. Developers should not need to use this constant unless they are writing applications for Mac OS X v10.2 and earlier.

Available in Mac OS X v10.3 and later.

Declared in `NSCursor.h`.

NSCustomImageRep Class Reference

Inherits from	NSImageRep : NSObject
Conforms to	NSCoding (NSImageRep) NSCopying (NSImageRep) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSCustomImageRep.h
Companion guide	Cocoa Drawing Guide
Related sample code	CompositeLab

Overview

An `NSCustomImageRep` object uses a delegate object to render an image. When called upon to produce an image, it sends a message to its delegate to do the actual drawing. You can use this class to support custom image formats without going to the trouble of subclassing `NSImageRep` directly.

Tasks

Initializing a New NSCustomImageRep

`- initWithDrawSelector:delegate:` (page 874)
Returns an `NSCustomImageRep` object initialized with the specified delegate information.

Identifying the Object

`- delegate` (page 874)
Returns the delegate object that renders the image for the receiver.

`- drawSelector` (page 874)
Returns the selector for the delegate's drawing method.

Instance Methods

delegate

Returns the delegate object that renders the image for the receiver.

```
- (id)delegate
```

Return Value
The delegate object.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSCustomImageRep.h`

drawSelector

Returns the selector for the delegate's drawing method.

```
- (SEL)drawSelector
```

Return Value
The selector for the delegate's drawing method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSCustomImageRep.h`

initWithDrawSelector:delegate:

Returns an `NSCustomImageRep` object initialized with the specified delegate information.

```
- (id)initWithDrawSelector:(SEL)aMethod delegate:(id)anObject
```

Parameters

aMethod

The selector to call when it is time to draw the image. The method should take a single parameter of type `id` that represents the `NSCustomImageRep` object that initiated drawing. The method must draw the image starting at the point (0, 0) in the current coordinate system.

anObject

The delegate object that responds to the selector in *aMethod*.

Return Value
An initialized `NSCustomImageRep` object, or `nil` if the object could not be initialized.

Discussion
When the receiver is asked to draw the image, it sends the specified message to the selector, passing itself as a parameter to the delegate method. The delegate's drawing method should have the following form:

```
- (void)myCustomDrawMethod:(id)anNSCustomImageRep;
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `draw` (page 1415) (NSImageRep)

Related Sample Code
CompositeLab

Declared In
`NSCustomImageRep.h`

NSDatePicker Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	AppKit/NSDatePicker.h
Related sample code	DatePicker Mountains ScriptingBridgeiCal

Overview

`NSDatePicker` is a subclass of `NSControl` that provides a user interface for displaying and editing an `NSDate` object.

`NSDatePicker` uses an `NSDatePickerCell` to implement much of the control's functionality. `NSDatePicker` provides cover methods for most of `NSDatePickerCell` methods, which invoke the corresponding cell method.

Tasks

Configuring Date Pickers

`- isBezeled` (page 882)
Returns whether the receiver has a bezeled border.

`- setBezeled:` (page 884)
Specifies whether the receiver draws a bezeled border.

`- isBordered` (page 883)
Returns whether the receiver has a plain border.

`- setBordered:` (page 885)
Specifies whether the receiver draws a plain border.

- `backgroundColor` (page 879)
  Returns the background color of the receiver.
- `setBackgroundColor:` (page 884)
  Sets the receiver's background color.
- `drawsBackground` (page 882)
  Returns whether the receiver draws the background.
- `setDrawsBackground:` (page 888)
  Specifies whether the receiver draws the background.
- `textColor` (page 891)
  Returns the text color of the receiver.
- `setTextColor:` (page 889)
  Sets the text color of the receiver.
- `datePickerStyle` (page 881)
  Returns the receiver's date picker style.
- `setDatePickerStyle:` (page 886)
  Sets the receiver's date picker style.
- `delegate` (page 881)
  Returns the delegate of the receiver's date picker cell.
- `setDelegate:` (page 887)
  Sets the delegate of the receiver's date picker cell.
- `datePickerElements` (page 880)
  Returns a bitmask that indicates which visual elements of the date picker are currently shown, and which won't be usable because they are hidden.
- `setDatePickerElements:` (page 886)
  Sets a bitmask that indicates which visual elements of the date picker are currently shown, and which won't be usable because they are hidden.

Controlling Date Picker Range and Mode

- `calendar` (page 880)
  Returns the calendar used by the receiver.
- `setCalendar:` (page 885)
  Sets the receiver's calendar.
- `locale` (page 883)
  Returns the receiver's locale.
- `setLocale:` (page 888)
  Sets the receiver's locale.
- `datePickerMode` (page 880)
  Returns the receiver's date picker mode.
- `setDatePickerMode:` (page 886)
  Sets the receiver's date picker mode.
- `timeZone` (page 891)
  Returns the receiver's time zone.

- `setTimeZone:` (page 890)
  Sets the receiver's time zone.

Accessing Object Values

- `dateValue` (page 881)
  Returns the receiver's date.
- `setDateValue:` (page 887)
  Sets the receiver's date to a new starting value.
- `timeInterval` (page 891)
  Returns the time interval that represents the date range.
- `setTimeInterval:` (page 890)
  Sets the time interval of the date range.

Constraining the Displayable/Selectable Range

- `minDate` (page 884)
  Returns the minimum date value the receiver allows as input.
- `setMinDate:` (page 889)
  Sets the minimum date allowed as input by the receiver.
- `maxDate` (page 883)
  Returns the maximum date value the receiver allows as input.
- `setMaxDate:` (page 889)
  Sets the maximum date allowed as input by the receiver.

Instance Methods

backgroundColor

Returns the background color of the receiver.

```
- (NSColor *)backgroundColor
```

Return Value
The background color of the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setBackgroundColor:` (page 884)

Declared In
`NSDatePicker.h`

calendar

Returns the calendar used by the receiver.

```
- (NSCalendar *)calendar
```

Return Value
The calendar used by the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setCalendar:` (page 885)

Declared In
`NSDatePicker.h`

datePickerElements

Returns a bitmask that indicates which visual elements of the date picker are currently shown, and which won't be usable because they are hidden.

```
- (NSDatePickerElementFlags)datePickerElements
```

Return Value
A bitmask that specifies the date picker elements displayed by the receiver. See “Constants” in `NSDatePickerCell` for a description of the possible values.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setDatePickerElements:` (page 886)

Related Sample Code
DatePicker

Declared In
`NSDatePicker.h`

datePickerMode

Returns the receiver’s date picker mode.

```
- (NSDatePickerMode)datePickerMode
```

Return Value
The receiver’s date picker mode.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDatePickerMode:` (page 886)

Declared In
`NSDatePicker.h`

datePickerStyle

Returns the receiver's date picker style.

```
- (NSDatePickerStyle)datePickerStyle
```

Return Value
The receiver's date picker style.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDatePickerStyle:` (page 886)

Related Sample Code
DatePicker

Declared In
`NSDatePicker.h`

dateValue

Returns the receiver's date.

```
- (NSDate *)dateValue
```

Return Value
The receiver's date.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDateValue:` (page 887)

Related Sample Code
DatePicker
ScriptingBridgeiCal

Declared In
`NSDatePicker.h`

delegate

Returns the delegate of the receiver's date picker cell.

```
- (id)delegate
```

Return Value
The delegate of the receiver's date picker cell.

Discussion
The date picker's `NSDatePickerCell` instance handles all delegate methods.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setDelegate:` (page 887)

Declared In
`NSDatePicker.h`

drawsBackground

Returns whether the receiver draws the background.

```
- (BOOL)drawsBackground
```

Return Value
`TRUE` if the receiver draws the background, `FALSE` otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setDrawsBackground:` (page 888)

Declared In
`NSDatePicker.h`

isBezeled

Returns whether the receiver has a bezeled border.

```
- (BOOL)isBezeled
```

Return Value
`TRUE` if the receiver has a bezeled border, `FALSE` otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setBezeled:` (page 884)

Declared In
`NSDatePicker.h`

isBordered

Returns whether the receiver has a plain border.

```
- (BOOL)isBordered
```

Return Value
`TRUE` if the receiver has a plain border, `FALSE` otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setBordered:` (page 885)

Declared In
`NSDatePicker.h`

locale

Returns the receiver's locale.

```
- (NSLocale *)locale
```

Return Value
The receiver's locale.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setLocale:` (page 888)

Declared In
`NSDatePicker.h`

maxDate

Returns the maximum date value the receiver allows as input.

```
- (NSDate *)maxDate
```

Return Value
The maximum date value the receiver allows as input. `nil` indicates no maximum date.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setMaxDate:` (page 889)

Declared In
`NSDatePicker.h`

minDate

Returns the minimum date value the receiver allows as input.

```
- (NSDate *)minDate
```

Return Value
The minimum date value the receiver allows as input. `nil` indicates no minimum date.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setMinDate:` (page 889)

Declared In
`NSDatePicker.h`

setBackgroundColor:

Sets the receiver's background color.

```
- (void)setBackgroundColor:(NSColor *)color
```

Parameters
color
The new background color.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- backgroundColor` (page 879)

Related Sample Code
DatePicker

Declared In
`NSDatePicker.h`

setBezeled:

Specifies whether the receiver draws a bezeled border.

```
- (void)setBezeled:(BOOL)flag
```

Parameters
flag
`TRUE` if the receiver has a bezeled border, `FALSE` otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- isBezeled (page 882)

Related Sample Code
DatePicker

Declared In
NSDatePicker.h

setBordered:

Specifies whether the receiver draws a plain border.

```
- (void)setBordered:(BOOL)flag
```

Parameters

flag

TRUE if the receiver has a plain border, FALSE otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- isBordered (page 883)

Related Sample Code
DatePicker

Declared In
NSDatePicker.h

setCalendar:

Sets the receiver's calendar.

```
- (void)setCalendar:(NSCalendar *)newCalendar
```

Parameters

newCalendar

The new calendar.

Availability
Available in Mac OS X v10.4 and later.

See Also
- calendar (page 880)

Declared In
NSDatePicker.h

setDatePickerElements:

Sets a bitmask that indicates which visual elements of the date picker are currently shown, and which won't be usable because they are hidden.

```
- (void)setDatePickerElements:(NSDatePickerElementFlags)elementFlags
```

Parameters

elementFlags

A bitmask that specifies the date picker elements displayed by the receiver. See "Constants" in `NSDatePickerCell` for a description of the possible values.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- datePickerElements` (page 880)

Related Sample Code
DatePicker

Declared In
`NSDatePicker.h`

setDatePickerMode:

Sets the receiver's date picker mode.

```
- (void)setDatePickerMode:(NSDatePickerMode)newMode
```

Parameters

newMode

The new date picker mode.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- datePickerMode` (page 880)

Related Sample Code
DatePicker

Declared In
`NSDatePicker.h`

setDatePickerStyle:

Sets the receiver's date picker style.

```
- (void)setDatePickerStyle:(NSDatePickerStyle)newStyle
```

Parameters

newStyle

The new date picker style.

Availability

Available in Mac OS X v10.4 and later.

See Also

- datePickerStyle (page 881)

Related Sample Code

DatePicker

Declared In

NSDatePicker.h

setDateValue:

Sets the receiver's date to a new starting value.

```
- (void)setDateValue:(NSDate *)newStartDate
```

Parameters

newStartDate

The new starting date.

Availability

Available in Mac OS X v10.4 and later.

See Also

- dateValue (page 881)

Related Sample Code

DatePicker

Declared In

NSDatePicker.h

setDelegate:

Sets the delegate of the receiver's date picker cell.

```
- (void)setDelegate:(id)anObject
```

Parameters

anObject

The new delegate.

Discussion

The date picker's NSDatePickerCell instance handles all delegate methods.

Availability

Available in Mac OS X v10.4 and later.

See Also
- delegate (page 881)

Related Sample Code
DatePicker

Declared In
NSDatePicker.h

setDrawsBackground:

Specifies whether the receiver draws the background.

```
- (void)setDrawsBackground:(BOOL)flag
```

Parameters

flag

TRUE if the receiver draws the background, FALSE otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- drawsBackground (page 882)

Related Sample Code
DatePicker

Declared In
NSDatePicker.h

setLocale:

Sets the receiver's locale.

```
- (void)setLocale:(NSLocale *)newLocale
```

Parameters

newLocale

The new locale.

Availability
Available in Mac OS X v10.4 and later.

See Also
- locale (page 883)

Declared In
NSDatePicker.h

setMaxDate:

Sets the maximum date allowed as input by the receiver.

```
- (void)setMaxDate:(NSDate *)date
```

Parameters

date

The maximum date allowed as input by the receiver. `nil` indicates no maximum date.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- maxDate` (page 883)

Related Sample Code
DatePicker

Declared In
`NSDatePicker.h`

setMinDate:

Sets the minimum date allowed as input by the receiver.

```
- (void)setMinDate:(NSDate *)date
```

Parameters

date

The minimum date allowed as input by the receiver. `nil` indicates no minimum date.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- minDate` (page 884)

Related Sample Code
DatePicker

Declared In
`NSDatePicker.h`

setTextColor:

Sets the text color of the receiver.

```
- (void)setTextColor:(NSColor *)color
```

Parameters

color

The new text color.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `textColor` (page 891)

Related Sample Code
DatePicker

Declared In
`NSDatePicker.h`

setTimeInterval:

Sets the time interval of the date range.

```
- (void)setTimeInterval:(NSTimeInterval)newTimeInterval
```

Parameters

newTimeInterval
The new time interval.

Discussion
The time interval only applies when the receiver is in the `NSRangeDateMode` mode.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `timeInterval` (page 891)

Declared In
`NSDatePicker.h`

setTimeZone:

Sets the receiver's time zone.

```
- (void)setTimeZone:(NSTimeZone *)newTimeZone
```

Parameters

newTimeZone
The new time zone.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `timeZone` (page 891)

Declared In
`NSDatePicker.h`

textColor

Returns the text color of the receiver.

```
- (NSColor *)textColor
```

Return Value
The text color of the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTextColor:` (page 889)

Declared In
`NSDatePicker.h`

timeInterval

Returns the time interval that represents the date range.

```
- (NSTimeInterval)timeInterval
```

Return Value
The time interval that represents the receiver's date range. The date range begins at the date returned by `dateValue` (page 881). This method returns 0 when the receiver is not in the `NSRangeDateMode` mode.

Special Considerations

Prior to Mac OS X v 10.5, this method always returned 0.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTimeInterval:` (page 890)

Declared In
`NSDatePicker.h`

timeZone

Returns the receiver's time zone.

```
- (NSTimeZone *)timeZone
```

Return Value
The receiver's time zone.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTimeZone:` (page 890)

Declared In
`NSDatePicker.h`

NSDatePickerCell Class Reference

Inherits from	NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	AppKit/NSDatePickerCell.h
Related sample code	DatePicker

Overview

An `NSDatePickerCell` instance controls the behavior of an `NSDatePicker` control, or of a single date picker cell in a matrix.

Tasks

Configuring Appearance

- `backgroundColor` (page 895)
  Returns the receiver's background color.
- `setBackgroundColor:` (page 899)
  Sets the receiver's background color
- `drawsBackground` (page 897)
  Returns whether the receiver draws the background.
- `setDrawsBackground:` (page 901)
  Sets whether the receiver draws the background.
- `textColor` (page 904)
  Returns the receiver's text color.
- `setTextColor:` (page 903)
  Sets the receiver's text color.

- `datePickerStyle` (page 896)
  Returns the receiver's date picker style.
- `setDatePickerStyle:` (page 900)
  Sets the receiver's date picker style.
- `datePickerElements` (page 896)
  Returns a bitmask that indicates which visual elements of the date picker are currently shown, and which won't be usable because they are hidden.
- `setDatePickerElements:` (page 900)
  Sets a bitmask that indicates which visual elements of the date picker are currently shown, and which won't be usable because they are hidden.

Range Mode

- `datePickerMode` (page 896)
  Returns the receiver's date picker mode.
- `setDatePickerMode:` (page 900)
  Sets the receiver's date picker mode.

Object Values

- `dateValue` (page 897)
  Returns the receiver's date.
- `setDateValue:` (page 901)
  Sets the receiver's date to a new starting value.
- `timeInterval` (page 904)
  Returns the time interval that represents the date range.
- `setTimeInterval:` (page 903)
  Sets the time interval of the date range.
- `calendar` (page 895)
  Returns the calendar used by the receiver.
- `setCalendar:` (page 899)
  Sets the receiver's calendar.
- `locale` (page 898)
  Returns the receiver's locale.
- `setLocale:` (page 902)
  Sets the receiver's locale.
- `timeZone` (page 905)
  Returns the receiver's time zone.
- `setTimeZone:` (page 904)
  Sets the receiver's time zone.

Date Range Constraints

- `minDate` (page 899)
  Returns the minimum date value that the receiver allows as input.
- `setMinDate:` (page 903)
  Sets the minimum date allowed as input by the receiver to the given date.
- `maxDate` (page 898)
  Returns the maximum date value that the receiver allows as input.
- `setMaxDate:` (page 902)
  Sets the maximum date allowed as input by the receiver to the given date.

Getting and Setting the Delegate

- `delegate` (page 897)
  Returns the receiver's delegate.
- `setDelegate:` (page 901)
  Sets the receiver's delegate.

Instance Methods

backgroundColor

Returns the receiver's background color.

```
- (NSColor *)backgroundColor
```

Return Value
The receiver's background color.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setBackgroundColor:` (page 899)

Declared In
`NSDatePickerCell.h`

calendar

Returns the calendar used by the receiver.

```
- (NSCalendar *)calendar
```

Return Value
The calendar used by the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setCalendar:` (page 899)

Declared In
`NSDatePickerCell.h`

datePickerElements

Returns a bitmask that indicates which visual elements of the date picker are currently shown, and which won't be usable because they are hidden.

```
- (NSDatePickerElementFlags)datePickerElements
```

Return Value
A bitmask that specifies the date picker elements displayed by the receiver. See "Constants" (page 905) for a description of the possible values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDatePickerElements:` (page 900)

Declared In
`NSDatePickerCell.h`

datePickerMode

Returns the receiver's date picker mode.

```
- (NSDatePickerMode)datePickerMode
```

Return Value
The receiver's date picker mode.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDatePickerMode:` (page 900)

Declared In
`NSDatePickerCell.h`

datePickerStyle

Returns the receiver's date picker style.

```
- (NSDatePickerStyle)datePickerStyle
```

Return Value
The receiver's date picker style.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDatePickerStyle:` (page 900)

Declared In
`NSDatePickerCell.h`

dateValue

Returns the receiver's date.

```
- (NSDate *)dateValue
```

Return Value
The receiver's date.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDateValue:` (page 901)

Declared In
`NSDatePickerCell.h`

delegate

Returns the receiver's delegate.

```
- (id < NSDatePickerCellDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDelegate:` (page 901)

Declared In
`NSDatePickerCell.h`

drawsBackground

Returns whether the receiver draws the background.

```
- (BOOL)drawsBackground
```

Return Value
`YES` if the receiver draws the background, otherwise `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setDrawsBackground:` (page 901)

Declared In
`NSDatePickerCell.h`

locale

Returns the receiver's locale.

```
- (NSLocale *)locale
```

Return Value
The receiver's locale.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setLocale:` (page 902)

Declared In
`NSDatePickerCell.h`

maxDate

Returns the maximum date value that the receiver allows as input.

```
- (NSDate *)maxDate
```

Return Value
The maximum date value that the receiver allows as input.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setMaxDate:` (page 902)

Declared In
`NSDatePickerCell.h`

minDate

Returns the minimum date value that the receiver allows as input.

```
- (NSDate *)minDate
```

Return Value
The minimum date value that the receiver allows as input.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setMinDate: (page 903)

Declared In
NSDatePickerCell.h

setBackgroundColor:

Sets the receiver's background color

```
- (void)setBackgroundColor:(NSColor *)color
```

Parameters
color
The new background color.

Availability
Available in Mac OS X v10.4 and later.

See Also
- backgroundColor (page 895)

Declared In
NSDatePickerCell.h

setCalendar:

Sets the receiver's calendar.

```
- (void)setCalendar:(NSCalendar *)newCalendar
```

Parameters
newCalendar
The calendar used by the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- backgroundColor (page 895)

Declared In
NSDatePickerCell.h

setDatePickerElements:

Sets a bitmask that indicates which visual elements of the date picker are currently shown, and which won't be usable because they are hidden.

```
- (void)setDatePickerElements:(NSDatePickerElementFlags)elementFlags
```

Parameters

elementFlags

A bitmask that specifies the date picker elements displayed by the receiver. See "Constants" (page 905) for a description of the possible values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- datePickerElements (page 896)

Declared In
NSDatePickerCell.h

setDatePickerMode:

Sets the receiver's date picker mode.

```
- (void)setDatePickerMode:(NSDatePickerMode)newMode
```

Parameters

newMode

The new date picker mode.

Availability
Available in Mac OS X v10.4 and later.

See Also
- datePickerMode (page 896)

Declared In
NSDatePickerCell.h

setDatePickerStyle:

Sets the receiver's date picker style.

```
- (void)setDatePickerStyle:(NSDatePickerStyle)newStyle
```

Parameters

newStyle

The new date picker style.

Availability
Available in Mac OS X v10.4 and later.

See Also
- datePickerStyle (page 896)

Declared In
NSDatePickerCell.h

setDateValue:

Sets the receiver's date to a new starting value.

```
- (void)setDateValue:(NSDate *)newStartDate
```

Parameters

newStartDate
The new starting date.

Availability
Available in Mac OS X v10.4 and later.

See Also
- dateValue (page 897)

Declared In
NSDatePickerCell.h

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSDatePickerCellDelegate >)anObject
```

Parameters

anObject
The receiver's delegate.

Availability
Available in Mac OS X v10.4 and later.

See Also
- delegate (page 897)

Declared In
NSDatePickerCell.h

setDrawsBackground:

Sets whether the receiver draws the background.

```
- (void)setDrawsBackground:(BOOL)flag
```

Parameters

flag

YES if the receiver draws the background, otherwise NO.

Availability
Available in Mac OS X v10.4 and later.

See Also
- drawsBackground (page 897)

Declared In
NSDatePickerCell.h

setLocale:

Sets the receiver's locale.

```
- (void)setLocale:(NSLocale *)newLocale
```

Parameters

newLocale

The receiver's locale.

Availability
Available in Mac OS X v10.4 and later.

See Also
- locale (page 898)

Declared In
NSDatePickerCell.h

setMaxDate:

Sets the maximum date allowed as input by the receiver to the given date.

```
- (void)setMaxDate:(NSDate *)date
```

Parameters

date

The maximum date the receiver allows as input. Pass nil to allow any date as the maximum value.

Availability
Available in Mac OS X v10.4 and later.

See Also
- maxDate (page 898)

Declared In
NSDatePickerCell.h

setMinDate:

Sets the minimum date allowed as input by the receiver to the given date.

```
- (void)setMinDate:(NSDate *)date
```

Parameters

date

The minimum date the receiver allows as input. Pass `nil` to allow any date as the minimum value.

Discussion

Passing `nil` for *date* allows any date as the minimum value.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `minDate` (page 899)

Declared In

`NSDatePickerCell.h`

setTextColor:

Sets the receiver's text color.

```
- (void)setTextColor:(NSColor *)color
```

Parameters

color

The new text color.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `textColor` (page 904)

Declared In

`NSDatePickerCell.h`

setTimeInterval:

Sets the time interval of the date range.

```
- (void)setTimeInterval:(NSTimeInterval)newTimeInterval
```

Parameters

newTimeInterval

The time interval of the date range.

Discussion

The time interval only applies when the receiver is in the `NSRangeDateMode` (page 907) mode.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `timeInterval` (page 904)

Declared In
`NSDatePickerCell.h`

setTimeZone:

Sets the receiver's time zone.

```
- (void)setTimeZone:(NSTimeZone *)newTimeZone
```

Parameters

newTimeZone
The receiver's time zone.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `timeZone` (page 905)

Declared In
`NSDatePickerCell.h`

textColor

Returns the receiver's text color.

```
- (NSColor *)textColor
```

Return Value
The receiver's text color.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTextColor:` (page 903)

Declared In
`NSDatePickerCell.h`

timeInterval

Returns the time interval that represents the date range.

```
- (NSTimeInterval)timeInterval
```

Return Value
The time interval that represents the date range.

Discussion
The date range begins at the date returned by `dateValue` (page 897). This method returns `0` when the receiver is not in the `NSRangeDateMode` (page 907) mode.

Special Considerations

Prior to Mac OS X v 10.5 this method always returned `0`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setTimeInterval:` (page 903)

Declared In
`NSDatePickerCell.h`

timeZone

Returns the receiver's time zone.

```
- (NSTimeZone *)timeZone
```

Return Value
The receiver's time zone.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setTimeZone:` (page 904)

Declared In
`NSDatePickerCell.h`

Constants

NSDatePickerStyle

Specifies a type for constants that define the visual appearance of the `NSDatePickerCell`.

```
typedef NSUInteger NSDatePickerStyle;
```

Discussion
For a discussion of possible values, see "Date Picker Style" (page 906).

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSDatePickerCell.h

Date Picker Style

The NSDatePickerStyle constants define the visual appearance of the NSDatePickerCell. These values are used by datePickerStyle (page 896) and setDatePickerStyle: (page 900).

```
enum {
    NSTextFieldAndStepperDatePickerStyle    = 0,
    NSClockAndCalendarDatePickerStyle       = 1,
    NSTextFieldDatePickerStyle              = 2
};
typedef NSUInteger NSDatePickerStyle;
```

Constants

NSTextFieldAndStepperDatePickerStyle
Provide a text field and stepper style interface.
Available in Mac OS X v10.4 and later.
Declared in NSDatePickerCell.h.

NSClockAndCalendarDatePickerStyle
Provide a visual clock and calendar style interface.
Available in Mac OS X v10.4 and later.
Declared in NSDatePickerCell.h.

NSTextFieldDatePickerStyle
Provide a text field interface.
Available in Mac OS X v10.5 and later.
Declared in NSDatePickerCell.h.

Declared In
NSDatePickerCell.h

NSDatePickerMode

Specifies a type for constants that define whether the control provides a single date, or a range of dates.

```
typedef NSUInteger NSDatePickerMode;
```

Discussion
For a discussion of possible values, see "Date Picker Mode" (page 907).

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSDatePickerCell.h

Date Picker Mode

The `NSDatePickerMode` constants define whether the control provides a single date, or a range of dates. These values are used by `datePickerMode` (page 896) and `setDatePickerMode:` (page 900).

```
enum {
    NSSingleDateMode = 0,
    NSRangeDateMode = 1
};
typedef NSUInteger NSDatePickerMode;
```

Constants

`NSSingleDateMode`

Allow selection of a single date.

Available in Mac OS X v10.4 and later.

Declared in `NSDatePickerCell.h`.

`NSRangeDateMode`

Allow selection of a range of dates. (First implemented in Mac OS X v 10.5.)

Available in Mac OS X v10.4 and later.

Declared in `NSDatePickerCell.h`.

Special Considerations

Prior to Mac OS X v 10.5 only `NSSingleDateMode` was implemented.

Declared In

`NSDatePickerCell.h`

NSDatePickerElementFlags

Specifies a type for constants that allow you to specify the date and time elements that the `NSDatePickerCell` can edit.

```
typedef NSUInteger NSDatePickerElementFlags;
```

Discussion

For a discussion of possible values, see "Date Picker Elements" (page 907).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSDatePickerCell.h`

Date Picker Elements

The `NSDatePickerElementFlag` constants allow you to specify the date and time elements that the `NSDatePickerCell` can edit by combining these constants using the C bitwise `OR` operator. These values are used by `datePickerElements` (page 896) and `setDatePickerElements:` (page 900):

```
enum {
    NSHourMinuteDatePickerElementFlag       = 0x000c,
    NSHourMinuteSecondDatePickerElementFlag = 0x000e,
    NSTimeZoneDatePickerElementFlag         = 0x0010,
    NSYearMonthDatePickerElementFlag        = 0x00c0,
    NSYearMonthDayDatePickerElementFlag     = 0x00e0,
    NSEraDatePickerElementFlag              = 0x0100,
};
typedef NSUInteger NSDatePickerElementFlags;
```

Constants

`NSHourMinuteDatePickerElementFlag`

Display and allow editing of the hour and minute elements of the date.

Available in Mac OS X v10.4 and later.

Declared in `NSDatePickerCell.h`.

`NSHourMinuteSecondDatePickerElementFlag`

Display and allow editing of the hour, minute and second elements of the date.

Available in Mac OS X v10.4 and later.

Declared in `NSDatePickerCell.h`.

`NSTimeZoneDatePickerElementFlag`

Display and allow editing of the time zone.

This flag has been declared for possible future use, and does not yet have any effect.

Available in Mac OS X v10.4 and later.

Declared in `NSDatePickerCell.h`.

`NSYearMonthDatePickerElementFlag`

Display and allow editing of the year and month elements of the date.

Available in Mac OS X v10.4 and later.

Declared in `NSDatePickerCell.h`.

`NSYearMonthDayDatePickerElementFlag`

Display and allow editing of the year, month and day elements of the date.

Available in Mac OS X v10.4 and later.

Declared in `NSDatePickerCell.h`.

`NSEraDatePickerElementFlag`

Display and allow editing of the era of the date, if applicable.

This flag has been declared for possible future use, and does not yet have any effect.

Available in Mac OS X v10.4 and later.

Declared in `NSDatePickerCell.h`.

Declared In

`NSDatePickerCell.h`

NSDictionaryController Class Reference

Inherits from	NSArrayController : NSObjectController : NSController : NSObject
Conforms to	NSCoding (NSController) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSDictionaryController.h
Availability	Available in Mac OS X v10.5 and later.
Companion guide	Cocoa Bindings Programming Topics
Related sample code	DictionaryController

Overview

`NSDictionaryController` is a bindings compatible class that manages display and editing of the contents of an `NSDictionary` object. `NSDictionaryController` transforms the contents of a dictionary into an array of key-value pairs that can be bound to user interface items such as the columns of an `NSTableView`.

The content of an `NSDictionaryController` instance is specified using the inherited method `setContent:` (page 1756) or by binding an `NSDictionary` instance to the `contentDictionary` binding. New key/value pairs inserted into the dictionary are created using the newObject method. The initial key name is set to the string returned by `initialKey` (page 912) (specified using `setInitialKey:` (page 915) or the `initialKey` binding). The initial value object is set to the object returned by `initialValue` (page 912) (specified using `setInitialValue:` (page 915) or the `initialValue` binding). The initial key name is copied to the newly inserted object, while the object returned by `initialValue` (page 912) is simply retained. As new items are inserted the controller enumerates the initial key name, resulting in key names such as "key", "key1", "key2", and so on. This behavior can be customized by overriding `newObject` (page 914).

An `NSDictionaryController` instance can be configured to exclude specified keys in a dictionary from being returned by `arrangedObjects` (page 911) using the `setExcludedKeys:` (page 914) method or by binding an array of key names to the `excludedKeys` binding. Similarly, you can specify an array of key names that are always included in the arranged objects, even if they are not present in the content dictionary, using the `setIncludedKeys:` (page 914) method or the `includedKeys` binding.

`NSDictionaryController` supports providing localized key names for the keys in the dictionary, allowing a user-friendly representation of the key name to be displayed. The localized key names are specified by a dictionary (using `setLocalizedKeyDictionary:` (page 915) or the `localizedKeyDictionary` binding) or by providing a strings table (using `setLocalizedKeyTable:` (page 916)).

The `arrangedObjects` (page 911) method returns an array of objects that implement the `NSDictionaryControllerKeyValuePair` informal protocol. User interface controls are bound to the arranged objects array using key paths such as: `arrangedObjects.key` (displays the key name), `arrangedObjects.value` (displays the value for the key), or `arrangedObjects.localizedKey` (displays the localized key name). See *NSDictionaryControllerKeyValuePair Protocol Reference* for more information.

Note: You must enable the "Validates Immediately" option for the value binding of all controls that edit the key names or values returned by `arrangedObjects` (page 911).

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Arranging Objects

- `arrangedObjects` (page 911)
  Returns an array containing the objects that represent the receiver's content.

Creating New Entries

- `newObject` (page 914)
  Creates and returns a new key-value pair to represent an entry in the content dictionary.

Localizing Key Names

- `setLocalizedKeyDictionary:` (page 915)
  Sets the localized key names that are displayed by the receiver in place of the key names.
- `localizedKeyDictionary` (page 913)
  Returns the receiver's localization dictionary.
- `localizedKeyTable` (page 913)
  Returns the strings file used to localize key names.
- `setLocalizedKeyTable:` (page 916)
  Specifies the strings file used to localize key names.

Keys to Display

- `setIncludedKeys:` (page 914)
  Sets the key names that are represented by a key-value pair, even if they are not present in the receiver's content dictionary.
- `includedKeys` (page 912)
  Returns an array containing the key names that are represented by a key-value pair, even if they are not present in the receiver's content dictionary.
- `setExcludedKeys:` (page 914)
  Sets the key names that are never displayed in the user interface items bound to the receiver.
- `excludedKeys` (page 911)
  Returns an array containing the key names that are never displayed in the user interface items bound to the receiver.

Setting Initial Key and Values

- `setInitialKey:` (page 915)
  Sets the string used as the initial key name for a newly inserted item.
- `initialKey` (page 912)
  Returns the string used as the initial key name for a newly inserted item.
- `setInitialValue:` (page 915)
  Sets the string used as the initial value for a newly inserted item.
- `initialValue` (page 912)
  Returns the string used as the initial value for a newly inserted item.

Instance Methods

arrangedObjects

Returns an array containing the objects that represent the receiver's content.

```
- (id)arrangedObjects
```

Return Value
An array of objects that implement the `NSDictionaryControllerKeyValuePair` informal protocol. See *NSDictionaryControllerKeyValuePair Protocol Reference* for more information.

excludedKeys

Returns an array containing the key names that are never displayed in the user interface items bound to the receiver.

```
- (NSArray *)excludedKeys
```

Return Value
An array containing the key names.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setExcludedKeys:` (page 914)

Declared In
`NSDictionaryController.h`

includedKeys

Returns an array containing the key names that are represented by a key-value pair, even if they are not present in the receiver's content dictionary.

```
- (NSArray *)includedKeys
```

Return Value
An array containing the key names.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setIncludedKeys:` (page 914)

Declared In
`NSDictionaryController.h`

initialKey

Returns the string used as the initial key name for a newly inserted item.

```
- (NSString *)initialKey
```

Return Value
The key name.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setInitialKey:` (page 915)

Declared In
`NSDictionaryController.h`

initialValue

Returns the string used as the initial value for a newly inserted item.

```
- (id)initialValue
```

Return Value
The value.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setInitialValue:` (page 915)

Declared In
`NSDictionaryController.h`

localizedKeyDictionary

Returns the receiver's localization dictionary.

```
- (NSDictionary *)localizedKeyDictionary
```

Return Value
A dictionary containing localized string values for the key names.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setLocalizedKeyDictionary:` (page 915)

Declared In
`NSDictionaryController.h`

localizedKeyTable

Returns the strings file used to localize key names.

```
- (NSString *)localizedKeyTable
```

Return Value
A string that specifies the string table to use when localizing key names.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setLocalizedKeyTable:` (page 916)

Declared In
`NSDictionaryController.h`

newObject

Creates and returns a new key-value pair to represent an entry in the content dictionary.

```
- (id)newObject
```

Return Value
An object that represents the key-value pair. The object must not be autoreleased, and must implement the NSDictionaryControllerKeyValuePair informal protocol

Discussion
This method is invoked for insertions of new key-value pairs, as well as transforming existing dictionary entries into key-value pairs for display. Objects returned by this method must implement the NSDictionaryControllerKeyValuePair informal protocol.

Special Considerations

Subclass implementations must ensure that the object returned by `newObject` is not autoreleased.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSDictionaryController.h`

setExcludedKeys:

Sets the key names that are never displayed in the user interface items bound to the receiver.

```
- (void)setExcludedKeys:(NSArray *)keys
```

Parameters
keys
An array containing the key names.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- excludedKeys` (page 911)

Declared In
`NSDictionaryController.h`

setIncludedKeys:

Sets the key names that are represented by a key-value pair, even if they are not present in the receiver's content dictionary.

```
- (void)setIncludedKeys:(NSArray *)keys
```

Parameters
keys
An array containing the key names.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `includedKeys` (page 912)

Declared In
`NSDictionaryController.h`

setInitialKey:

Sets the string used as the initial key name for a newly inserted item.

```
- (void)setInitialKey:(NSString *)key
```

Parameters

key
The key name. The string is copied by the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `initialKey` (page 912)

Declared In
`NSDictionaryController.h`

setInitialValue:

Sets the string used as the initial value for a newly inserted item.

```
- (void)setInitialValue:(id)value
```

Parameters

value
The initial value.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `initialValue` (page 912)

Declared In
`NSDictionaryController.h`

setLocalizedKeyDictionary:

Sets the localized key names that are displayed by the receiver in place of the key names.

```
- (void)setLocalizedKeyDictionary:(NSDictionary *)dictionary
```

Parameters

dictionary

A dictionary containing the localized key name strings.

Discussion

The dictionary contains the key names as the keys, and the localized key names as the corresponding values.

Availability

Available in Mac OS X v10.5 and later.

See Also

- localizedKeyDictionary (page 913)

Declared In

NSDictionaryController.h

setLocalizedKeyTable:

Specifies the strings file used to localize key names.

```
- (void)setLocalizedKeyTable:(NSString *)stringsFile
```

Parameters

stringsFile

Specifies the string table to use when localizing key names.

Discussion

The string table must reside within the application's resource . See Strings Files in Introduction to Internationalization Programming Topics.

Availability

Available in Mac OS X v10.5 and later.

See Also

- setLocalizedKeyDictionary: (page 915)- localizedKeyDictionary (page 913)

Declared In

NSDictionaryController.h

Constants

Exposed Bindings

The following constants are used to specify a binding to bind:toObject:withKeyPath:options: (page 3699), infoForBinding: (page 3700), unbind: (page 3701), and valueClassForBinding: (page 3701). See the *Cocoa Bindings Reference* for more information.

```
NSString *NSContentDictionaryBinding;
NSString *NSIncludedKeysBinding;
NSString *NSExcludedKeysBinding;
NSString *NSLocalizedKeyDictionaryBinding;
NSString *NSInitialKeyBinding;
NSString *NSInitialValueBinding;
```

Constants

`NSContentDictionaryBinding`

A dictionary used as the content dictionary.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSIncludedKeysBinding`

An array containing the key-value pairs always represented by the receiver.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSExcludedKeysBinding`

An array containing the key names that are never displayed in the user interface items bound to the receiver.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSLocalizedKeyDictionaryBinding`

A dictionary containing the localized key names that are displayed by the receiver in place of the key names.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSInitialKeyBinding`

A string used as the initial key name for newly inserted items.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSInitialValueBinding`

A string used as the initial value for newly inserted items.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

Declared In

`AppKit/NSDictionaryController.h`

NSDockTile Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSDockTile.h
Companion guide	Dock Tile Programming Guide
Related sample code	DockTile

Overview

The `NSDockTile` class lets you customize the visual representation for your application's miniaturized windows and application icon as they appear in the Dock. You do not create Dock tile objects explicitly in your application. Instead, you retrieve the Dock tile for an existing window or for the application by calling that object's `dockTile` method.

Typically, you do not subclass the `NSDockTile` class. Instead, you use the methods of the class to make the following customizations:

- Badge the tile with a custom string.
- Remove or show the application icon badge.
- Draw the tile content yourself.

If you decide to draw the tile content yourself, you must provide a custom content view to handle the drawing.

Application Dock Tiles

An application Dock tile defaults to display the application's `applicationIconImage` (page 138).

The application Dock tile never shows a smaller application icon badge.

Whether using the default or custom view, the application Dock tile may be badged with a short custom string.

Window Dock Tiles

A window Dock tile defaults to display a miniaturized version of the windows contents with a badge derived from the application Dock icon, including any customized application Dock icon. The default window Dock tile image may not be badged with a custom string.

A window Dock tile can use a custom view to draw the Dock icon. If a custom view is used, no application badge will be added, but the text label will be overlaid on top of the icon.

Tasks

Drawing the Tile's Content

- `setContentView:` (page 923)
  Sets the view to use for drawing the dock tile contents.
- `contentView` (page 921)
  Returns the view used to draw the dock tile contents.

Getting the Tile Information

- `size` (page 924)
  Returns the size of the tile.
- `owner` (page 922)
  Returns the object represented by the dock tile.

Applying Badge Icons to the Tile

- `setShowsApplicationBadge:` (page 923)
  Sets whether the tile should be badged with the application's icon.
- `showsApplicationBadge` (page 924)
  Returns a Boolean value indicating whether the tile is badged with the application's icon.
- `setBadgeLabel:` (page 922)
  Sets the string to be displayed in the tile's badging area.
- `badgeLabel` (page 921)
  Returns the tile's current badge label.

Updating the Dock Tile

- `display` (page 921)
  Redraws the dock tile's content.

Instance Methods

badgeLabel

Returns the tile's current badge label.

```
- (NSString *)badgeLabel
```

Return Value
The localized string to be displayed in the tile's badging area. This string may be empty or `nil`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setBadgeLabel:` (page 922)

Declared In
`NSDockTile.h`

contentView

Returns the view used to draw the dock tile contents.

```
- (NSView *)contentView
```

Return Value
The view used to draw the tile.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setContentView:` (page 923)

Declared In
`NSDockTile.h`

display

Redraws the dock tile's content.

```
- (void)display
```

Discussion
If a custom content view is provided, Cocoa calls the `drawRect:` method of that view (and its subviews) to draw the tile's content.

You can call this method to force the redrawing of the dock tile contents. You might do this if the contents of the underlying application or window change in a way that would require a refreshing of the tile. Some types of system activity, such as resizing the dock, may trigger automatic redraws of the tile. In most cases, however, your application is responsible for triggering redraws.

Cocoa does not automatically redraw the contents of your dock tile. Instead, your application must explicitly send `display` messages to the dock tile object whenever the contents of your view change and need to be redrawn.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
DockTile

Declared In
`NSDockTile.h`

owner

Returns the object represented by the dock tile.

```
- (id)owner
```

Return Value
The object represented by the dock tile. This is either the `NSApplication` object or one of your application's `NSWindow` objects.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSDockTile.h`

setBadgeLabel:

Sets the string to be displayed in the tile's badging area.

```
- (void)setBadgeLabel:(NSString *)string
```

Parameters

string

The localized string to display. This string can contain a count value or other badging information. To clear the badge string, specify an empty string (`@""`) or `nil`.

Discussion
The appearance of the badge area is system defined.

Window dock tiles only display a badge label when there is a custom view associated with the dock tile.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `badgeLabel` (page 921)
- `showsApplicationBadge` (page 924)

Declared In
`NSDockTile.h`

setContentView:

Sets the view to use for drawing the dock tile contents.

```
- (void)setContentView:(NSView *)view
```

Parameters

view

The view to use for drawing the tile. This view may contain additional subviews.

Discussion
The view you specify should be height and width resizable.

Cocoa does not automatically redraw the contents of your dock tile. Instead, your application must explicitly send display messages to the dock tile object whenever the contents of your view change and need to be redrawn. Your dock tile view is responsible for drawing the entire contents of the dock tile. Your view does not need to draw the application or custom string badges.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `contentView` (page 921)
- `display` (page 921)

Related Sample Code
DockTile

Declared In
`NSDockTile.h`

setShowsApplicationBadge:

Sets whether the tile should be badged with the application's icon.

```
- (void)setShowsApplicationBadge:(BOOL)flag
```

Parameters

flag

`YES` to show the application icon; otherwise, `NO` to hide it.

Discussion
Miniaturized windows include the application badge by default to convey the associated application to the user. In Mac OS X v10.5 and later, application tiles do not support the application badge. A miniaturized window with a custom view does not draw the application badge.

The application icon is positioned automatically in the tile by the `NSDockTile` object.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- showsApplicationBadge` (page 924)

Declared In
`NSDockTile.h`

showsApplicationBadge

Returns a Boolean value indicating whether the tile is badged with the application's icon.

```
- (BOOL)showsApplicationBadge
```

Return Value
`YES` if the tile is badged; otherwise, `NO`. Returns `YES` by default.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setShowsApplicationBadge:` (page 923)

Declared In
`NSDockTile.h`

size

Returns the size of the tile.

```
- (NSSize)size
```

Return Value
The size of the tile, measured in screen coordinates.

Discussion
The size returned by this method corresponds to the size of the backing store in the dock, which may be bigger than the actual tile displayed on the screen.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
DockTile

Declared In
`NSDockTile.h`

Constants

Dock Tile Plug-In Support Version

The version of the AppKit framework containing support for dock tile plug-ins.

```
#define NSAppKitVersionNumberWithDockTilePlugInSupport 1001.0
```

Constants

`NSAppKitVersionNumberWithDockTilePlugInSupport`

The specific version of the AppKit framework that introduced support for dock tile plug-ins.. Developers should not need to use this constant unless they are writing applications for Mac OS X v10.5 and earlier.

Available in Mac OS X v10.6 and later.

Declared in `NSDockTile.h`.

NSDocument Class Reference

Inherits from	NSObject
Conforms to	NSUserInterfaceValidations NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSDocument.h AppKit/NSDocumentScripting.h
Companion guide	Document-Based Applications Overview
Related sample code	ImageApp iSpend iSpendPlugin QTAudioExtractionPanel Simple Bindings Adoption

Class at a Glance

`NSDocument` is an abstract class that defines the interface for documents, objects that can internally represent data displayed in windows and that can read data from and write data to files. Documents create and manage one or more window controllers and are in turn managed by a document controller. Documents respond to first-responder action messages to save, revert, and print their data.

Principal Attributes

- Window controllers
- Filenames
- Document types
- Print information

`init` (page 954)
Designated initializer for new documents

`initWithContentsOfURL:ofType:error:` (page 957)
For existing documents

Commonly Used Methods

`dataOfType:error:` (page 944)
Returns the document's data in a specified type.

`readFromData:ofType:error:` (page 967)
Sets the contents of this document by reading from data of a specified type.

`writeToURL:ofType:error:` (page 996)
Writes the document's data to a URL.

`readFromURL:ofType:error:` (page 970)
Reads the document's data from a file.

`windowNibName` (page 992)
Returns the name of the document's sole nib file (resulting in the creation of a window controller for the window in that file).

`makeWindowControllers` (page 961)
Creates and returns the window controllers used to manage document windows.

Overview

`NSDocument` is an abstract class that defines the interface for documents.

Conceptually, a document is a container for a body of information identified by a name under which it is stored in a disk file. In this sense, however, the document is not the same as the file but is an object in memory that owns and manages the document data. In the context of the Application Kit, a document is an instance of a custom `NSDocument` subclass that knows how to represent internally, in one or more formats, persistent data that is displayed in windows.

A document can read that data from a file and write it to a file. It is also the first-responder target for many menu commands related to documents, such as Save, Revert, and Print. A document manages its window's edited status and is set up to perform undo and redo operations. When a window is closing, the document is asked before the window delegate to approve the closing.

`NSDocument` is one of the triad of Application Kit classes that establish an architectural basis for document-based applications (the others being `NSDocumentController` and `NSWindowController`).

Subclassing NSDocument

`NSDocument` is designed to be subclassed. That is, `NSDocument` is an abstract class, and your application must create at least one `NSDocument` subclass to use the document architecture. To create a useful `NSDocument` subclass, you must override some methods, and you can optionally override others.

The `NSDocument` class itself knows how to handle document data as undifferentiated lumps; although it understands that these lumps are typed, it knows nothing about particular types. In their overrides of the data-based reading and writing methods, subclasses must add the knowledge of particular types and how data of the document's native type is structured internally. Subclasses are also responsible for the creation of the window controllers that manage document windows and for the implementation of undo and redo. The `NSDocument` class takes care of much of the rest, including generally managing the state of the document.

See "Creating a Subclass of NSDocument" in *Document-Based Applications Overview* for more information about creating subclasses of `NSDocument`, particularly the list of primitive methods that subclasses must override and those that you can optionally override.

Writing of HFS Creator and File Type Codes

The `fileAttributesToWriteToFile:ofType:saveOperation:` (page 946) method can be overridden to specify that a creator code or file type code (or both) should be written to a file as it is being saved. See `NSFileManager` for descriptions of the `NSFileHFSCreatorCode` and `NSFileHFSTypeCode` file attributes. The `NSDocument` implementation of `fileAttributesToWriteToFile:ofType:saveOperation:` returns zeroed-out creator and file type codes, effectively excluding creator code and file type code from the attribute preservation described in `fileAttributesToWriteToFile:ofType:saveOperation:`.

NSDocument Saving Behavior

`NSDocument` implements document saving in a way that preserves, when possible, various attributes of each document, including:

- Creation date
- Permissions/privileges
- Location of the document's icon in its parent folder's Icon View Finder window
- Value of the document's Show Extension setting

Care is also taken to save documents in a way that does not break any user-created aliases that may point to documents. As a result, some methods in any class of `NSDocument` may be invoked with parameters that do not have the same meaning as they did in early releases of Mac OS X. It is important that overrides of `writeToURL:ofType:error:` (page 996) and `writeToURL:ofType:forSaveOperation:originalContentsURL:error:` (page 997) make no assumptions about the file paths passed as parameters, including:

- The location to which the file is being written. This location might be a hidden temporary directory.
- The name of the file being written. It is possible that this file has no obvious relation to the document name.
- The relation of any file being passed, including the original file, to the return value of `fileName` (page 948).

When updating your application to link against Mac OS X v10.5, keep in mind that it's usually more appropriate to invoke in your application code one of the `NSDocument save...` methods than one of the `write...` methods. The `write...` methods are there primarily for you to override. The `saveToURL:ofType:forSaveOperation:error:` (page 980) method that's meant always to be invoked during document saving, invokes `setFileModificationDate:` (page 981) with the file's new modification date after it has been written (for `NSSaveOperation` (page 999) and `NSSaveAsOperation` (page 999) only).

Likewise, it's usually more appropriate to invoke in your application code one of the `NSDocument` `revert...` methods than one of the `read...` methods. The `read...` methods are there primarily for you to override. The `revertToContentsOfURL:ofType:error:` (page 971) method that's meant always to be invoked during rereading of an open document, invokes `setFileModificationDate:` (page 981) with the file's modification date after it has been read.

Multicore Considerations

In Mac OS X v10.6 and later, `NSDocument` supports the ability to open multiple documents concurrently. However, this support requires the cooperation of the document object. If your document subclass is able to read specific document types independently of other similar documents, you should override the `canConcurrentlyReadDocumentsOfType:` (page 938) class method and return `YES` for the appropriate document types. If specific document types rely on shared state information, however, you should return `NO` for those types.

Tasks

Initializing

- `init` (page 954)
  Initializes and returns an empty `NSDocument` object.
- `initWithContentsOfURL:ofType:error:` (page 957)
  Initializes a document located by a URL of a specified type.
- `initForURL:withContentsOfURL:ofType:error:` (page 955)
  Initializes a document located by a URL of a specified type, but by reading the contents for the document from a different URL.
- `initWithType:error:` (page 957)
  Initializes a document of a specified type.

Loading Document Data

- `dataOfType:error:` (page 944)
  Creates and returns a data object that contains the contents of the document, formatted to a specified type.
- `fileWrapperOfType:error:` (page 951)
  Creates and returns a file wrapper that contains the contents of the document, formatted to the specified type.
- `readFromData:ofType:error:` (page 967)
  Sets the contents of this document by reading from data of a specified type and returns `YES` if successful.

Creating and Managing Window Controllers

- `makeWindowControllers` (page 961)
  Subclasses may override this method to create the initial window controller(s) for the document.
- `windowNibName` (page 992)
  Overridden by subclasses to return the name of the document's sole nib file.
- `windowControllerDidLoadNib:` (page 990)
  Sent after the specified window controller loads a nib file if the receiver is the nib file's owner.
- `windowControllerWillLoadNib:` (page 991)
  Sent before the specified window controller loads a nib file if the receiver is the nib file's owner.
- `windowControllers` (page 991)
  Returns the receiver's current window controllers.
- `addWindowController:` (page 940)
  Adds the specified window controller to the array of window controllers associated with the receiver.
- `removeWindowController:` (page 970)
  Removes the specified window controller from the receiver's array of window controllers.
- `shouldCloseWindowController:delegate:shouldCloseSelector:contextInfo:` (page 986)
  Invokes *`shouldCloseSelector`* with the result of `canCloseDocumentWithDelegate:shouldCloseSelector:contextInfo:` (page 943) if the the specified window controller that is closing is the last one or is marked as causing the document to close.

Managing Document Windows

- `showWindows` (page 987)
  Displays all of the document's windows, bringing them to the front and making them main or key as necessary.
- `displayName` (page 945)
  Returns the name of the receiver as displayed in the title bars of the document's windows and in alert dialogs related to the document.
- `setWindow:` (page 985)
  Sets the `window` Interface Builder outlet of this class.
- `windowForSheet` (page 992)
  Returns the most appropriate window, of the windows associated with the receiver, to use as the parent window of a document-modal sheet.

Reading From and Writing to Files

- `readFromFileWrapper:ofType:error:` (page 969)
  Sets the contents of this document by reading from a file wrapper of a specified type.
- `fileModificationDate` (page 948)
  Returns the last known modification date of the document's on-disk representation.
- `setFileModificationDate:` (page 981)
  Sets the last known modification date of the document's on-disk representation to the given modification date.

- `runModalSavePanelForSaveOperation:delegate:didSaveSelector:contextInfo:` (page 975)
  Presents a modal Save panel to the user, then tries to save the document if the user approves the panel.
- `shouldRunSavePanelWithAccessoryView` (page 987)
  Returns `YES` by default; as a result, when `NSDocument` displays the Save panel, it includes an accessory view containing a pop-up menu of supported writable document types.
- `keepBackupFile` (page 958)
  Returns a Boolean value indicating whether the receiver should keep the backup files created before document data is written to a file (`NO` by default).

Reading From and Writing to URLs

- `readFromURL:ofType:error:` (page 970)
  Sets the contents of this document by reading from a file or file package, of a specified type, located by a URL.
- `writeToURL:ofType:error:` (page 996)
  Writes the contents of the document to a file or file package located by a URL, formatted to a specified type.
- `writeSafelyToURL:ofType:forSaveOperation:error:` (page 994)
  Writes the contents of the document to a file or file package located by a URL.
- `writeToURL:ofType:forSaveOperation:originalContentsURL:error:` (page 997)
  Writes the contents of the document to a file or file package located by a URL.
- `setFileURL:` (page 982)
  Sets the location of the document's on-disk representation.
- `fileURL` (page 951)
  Returns the location of the document's on-disk representation.
- `fileAttributesToWriteToURL:ofType:forSaveOperation:originalContentsURL:error:` (page 946)
  As a file is being saved, returns the attributes that should be written to a file or file package located by a URL, formatted to a specified type, for a particular kind of save operation.
- `saveToURL:ofType:forSaveOperation:delegate:didSaveSelector:contextInfo:` (page 979)
  Saves the contents of the document to a file or file package located by a URL, formatted to a specified type, for a particular kind of save operation.
- `saveToURL:ofType:forSaveOperation:error:` (page 980)
  Saves the contents of the document to a file or file package located by a URL, formatted to a specified type, for a particular kind of save operation, and returns `YES` if successful.

Autosaving

- `hasUnautosavedChanges` (page 954)
  Return `YES` if the document has changes that have not been autosaved, as determined by the history of previous invocations of `updateChangeCount:` (page 988).
- `autosaveDocumentWithDelegate:didAutosaveSelector:contextInfo:` (page 941)
  Autosaves the document's contents at an appropriate location.

- `autosavingFileType` (page 942)
  Returns the document type that should be used for an autosave operation.
- `setAutosavedContentsFileURL:` (page 980)
  Sets the location of the most recently autosaved document contents.
- `autosavedContentsFileURL` (page 941)
  Returns the location of the most recently autosaved document contents.

Managing Document Status

- `isDocumentEdited` (page 958)
  Returns `YES` if the receiver has changes that have not been saved, `NO` otherwise.
- `updateChangeCount:` (page 988)
  Updates the receiver's change count according to the given change type.
- `fileNameExtensionWasHiddenInLastRunSavePanel` (page 949)
  Returns `YES` if a Save panel was presented by this document and the user chose to hide the name extension of the file that was selected in that Save panel.

Handling User Actions

- `prepareSavePanel:` (page 962)
  Invoked by `runModalSavePanelForSaveOperation:delegate:didSaveSelector:contextInfo:` (page 975) to do any customization of the given Save panel.
- `printDocument:` (page 964)
  Prints the receiver in response to the user choosing the Print menu command.
- `runPageLayout:` (page 975)
  The action method invoked in the receiver as first responder when the user chooses the Page Setup menu command.
- `revertDocumentToSaved:` (page 971)
  The action of the File menu item Revert in a document-based application.
- `saveDocument:` (page 976)
  The action method invoked in the receiver as first responder when the user chooses the Save menu command.
- `saveDocumentAs:` (page 977)
  The action method invoked in the receiver as first responder when the user chooses the Save As menu command.
- `saveDocumentTo:` (page 977)
  The action method invoked in the receiver as first responder when the user chooses the Save To menu command.
- `saveDocumentWithDelegate:didSaveSelector:contextInfo:` (page 978)
  Saves the document.

Closing Documents

- `canCloseDocumentWithDelegate:shouldCloseSelector:contextInfo:` (page 943)
  If the receiver is not dirty, this method immediately calls the *`shouldCloseSelector`* callback on the specified delegate with `YES`.
- `close` (page 943)
  Closes all windows owned by the receiver and removes the receiver from the list of documents maintained by the document controller, which consequently releases it.

Reverting Documents

- `revertToContentsOfURL:ofType:error:` (page 971)
  Discards all unsaved document modifications and replaces the document's contents by reading a file or file package located by a URL of a specified type.

Printing Documents

- `printInfo` (page 966)
  Returns the receiver's customized `NSPrintInfo` object or the default `NSPrintInfo` instance.
- `setPrintInfo:` (page 984)
  Sets the receiver's `NSPrintInfo` object.
- `preparePageLayout:` (page 962)
  Invoked by `runModalPageLayoutWithPrintInfo:` (page 973) and `runModalPageLayoutWithPrintInfo:delegate:didRunSelector:contextInfo:` (page 973) to do any customization of the Page Layout panel *`pageLayout`*, such as adding an accessory view.
- `runModalPageLayoutWithPrintInfo:delegate:didRunSelector:contextInfo:` (page 973)
  Runs the modal page layout panel with the receiver's printing information object
- `runModalPrintOperation:delegate:didRunSelector:contextInfo:` (page 974)
  Runs the specified print operation modally.
- `shouldChangePrintInfo:` (page 985)
  Returns a Boolean value indicating whether the receiver should allow changes to the default `NSPrintInfo` object used in printing the document.
- `printDocumentWithSettings:showPrintPanel:delegate:didPrintSelector:contextInfo:` (page 965)
  Prints the document.
- `printOperationWithSettings:error:` (page 966)
  Creates a print operation and returns it if successful.

Handling Errors

- `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 963)
  Presents an error alert to the user as a modal panel.
- `presentError:` (page 962)
  Presents an error alert to the user as a modal panel.

- `willPresentError:` (page 989)
  Called when the receiver is about to present an error.

Working with Undo Manager

- `hasUndoManager` (page 954)
  Returns a Boolean value indicating whether the receiver owns or should own an `NSUndoManager` object.
- `setHasUndoManager:` (page 983)
  Sets whether the receiver has its own `NSUndoManager` object.
- `setUndoManager:` (page 984)
  Sets the undo manager owned by the receiver to the specified undo manager and releases any undo manager currently owned by the receiver.
- `undoManager` (page 988)
  Returns the receiver's undo manager.

Managing File Types

- `setFileType:` (page 982)
  Sets the document type under which the file is saved.
- `fileType` (page 950)
  Returns the document type under which the receiver is saved.
- `fileTypeFromLastRunSavePanel` (page 950)
  Returns the file type that was last selected in the Save panel.
+ `isNativeType:` (page 939)
  Returns a Boolean value indicating whether document data of the specified type is a native type—one the receiver can both read and write.
+ `readableTypes` (page 939)
  Returns the types of data the receiver can read natively and any types filterable to that native type.
+ `canConcurrentlyReadDocumentsOfType:` (page 938)
  Returns a Boolean value indicating whether the receiver reads multiple documents of the given type concurrently.
+ `writableTypes` (page 940)
  Returns the types of data the receiver can write natively and any types filterable to that native type.
- `writableTypesForSaveOperation:` (page 993)
  Returns the names of the types to which this document can be saved for a specified kind of save operation.
- `fileNameExtensionForType:saveOperation:` (page 949)
  Returns a filename extension that can be appended to a base filename, for a specified file type and kind of save operation.

Validating User Interface Items

- `validateUserInterfaceItem:` (page 989)
  Validates the specified user interface item that the receiver manages.

Scripting

- `handleCloseScriptCommand:` (page 952)
  Handles the Close AppleScript command by attempting to close the document.
- `handlePrintScriptCommand:` (page 953)
  Handles the Print AppleScript command by attempting to print the document.
- `handleSaveScriptCommand:` (page 953)
  Handles the Save AppleScript command by attempting to save the document.
- `objectSpecifier` (page 961)
  Returns an object specifier for the document.
- `lastComponentOfFileName` (page 959)
  Returns the document name in terms of the scripting name property (the name a script writer would use to specify the document in a script).
- `setLastComponentOfFileName:` (page 983)
  Sets the document name to the given string in terms of the scripting name property (the name a script writer would use to specify the document in a script).

Deprecated Methods

- `canCloseDocument` (page 942) Available in Mac OS X v10.0 through Mac OS X v10.3
  This method is no longer supported. (Deprecated. Use `canCloseDocumentWithDelegate:shouldCloseSelector:contextInfo:` (page 943) instead.)
- `fileNameFromRunningSavePanelForSaveOperation:` (page 950) Available in Mac OS X v10.0 through Mac OS X v10.3
  Returns the filename entered into the Save panel. (Deprecated. Use `saveDocumentWithDelegate:didSaveSelector:contextInfo:` (page 978) instead.)
- `shouldCloseWindowController:` (page 986) Available in Mac OS X v10.0 through Mac OS X v10.3
  Gives the user an opportunity to save the document. (Deprecated. Use `shouldCloseWindowController:delegate:shouldCloseSelector:contextInfo:` (page 986) instead.)
- `validateMenuItem:` (page 989) Available in Mac OS X v10.0 through Mac OS X v10.3
  Validates the Revert menu item and items selected from the Save panel's pop-up list of writable document types items. (Deprecated. Use `validateUserInterfaceItem:` (page 989) instead.)
- `dataRepresentationOfType:` (page 945) Deprecated in Mac OS X v10.4
  A primitive method overridden by subclasses to return a data object that represents the data of the receiver in a given type. (Deprecated. Use `dataOfType:error:` (page 944) instead.)
- `fileAttributesToWriteToFile:ofType:saveOperation:` (page 946) Deprecated in Mac OS X v10.4
  Returns the file attributes that should be written to the named document file of the specified type. (Deprecated. Use

`fileAttributesToWriteToURL:ofType:forSaveOperation:originalContentsURL:error:` (page 946) instead.)

- `fileName` (page 948) Deprecated in Mac OS X v10.4
  Returns the filename (as a fully qualified path) under which the receiver has been saved. (Deprecated. Use `fileURL` (page 951) instead.)
- `fileWrapperRepresentationOfType:` (page 952) Deprecated in Mac OS X v10.4
  Returns an NSFileWrapper object that represents the data of the receiver in a given type. (Deprecated. Use `fileWrapperOfType:error:` (page 951) instead.)
- `initWithContentsOfFile:ofType:` (page 956) Deprecated in Mac OS X v10.4
  Initializes and returns an `NSDocument` object. (Deprecated. Use `initWithContentsOfURL:ofType:error:` (page 957) instead.)
- `initWithContentsOfURL:ofType:` (page 956) Deprecated in Mac OS X v10.4
  Initializes and returns an NSDocument object of a given document type. (Deprecated. Use `initWithContentsOfURL:ofType:error:` (page 957) instead.)
- `loadDataRepresentation:ofType:` (page 959) Deprecated in Mac OS X v10.4
  Overridden by subclasses to load document data. (Deprecated. Use `readFromData:ofType:error:` (page 967) instead.)
- `loadFileWrapperRepresentation:ofType:` (page 960) Deprecated in Mac OS X v10.4
  Loads document data from a given file wrapper. (Deprecated. Use `readFromFileWrapper:ofType:error:` (page 969) instead.)
- `printShowingPrintPanel:` (page 967) Deprecated in Mac OS X v10.4
  Overridden by subclasses to print the current document's (the receiver's) data. (Deprecated. Use `printDocumentWithSettings:showPrintPanel:delegate:didPrintSelector:contextInfo:` (page 965) instead.)
- `readFromFile:ofType:` (page 968) Deprecated in Mac OS X v10.4
  Reads and loads document data of the given type from the given file. (Deprecated. Use `readFromURL:ofType:error:` (page 970) instead.)
- `readFromURL:ofType:` (page 969) Deprecated in Mac OS X v10.4
  Reads and loads document data. (Deprecated. Use `readFromURL:ofType:error:` (page 970) instead.)
- `revertToSavedFromFile:ofType:` (page 972) Deprecated in Mac OS X v10.4
  Reverts the receiver to the data stored in the file system. (Deprecated. Use `revertToContentsOfURL:ofType:error:` (page 971) instead.)
- `revertToSavedFromURL:ofType:` (page 972) Deprecated in Mac OS X v10.4
  Reverts the receiver. (Deprecated. Use `revertToContentsOfURL:ofType:error:` (page 971) instead.)
- `runModalPageLayoutWithPrintInfo:` (page 973) Deprecated in Mac OS X v10.4
  Runs the page layout modal panel with the receiver's printing information object. (Deprecated. Use `runModalPageLayoutWithPrintInfo:delegate:didRunSelector:contextInfo:` (page 973) instead.)
- `saveToFile:saveOperation:delegate:didSaveSelector:contextInfo:` (page 978) Deprecated in Mac OS X v10.4
  Called after the user has been given the opportunity to select a destination through the modal Save panel. (Deprecated. Use `saveToURL:ofType:forSaveOperation:delegate:didSaveSelector:contextInfo:` (page 979) instead.)

- `setFileName:` (page 981) Deprecated in Mac OS X v10.4
  Sets the file (filename and directory path) under which document data is saved. (Deprecated. Use `setFileURL:` (page 982) instead.)
- `writeToFile:ofType:` (page 995) Deprecated in Mac OS X v10.4
  Writes document data to a file. (Deprecated. Use `writeToURL:ofType:error:` (page 996) instead.)
- `writeToFile:ofType:originalFile:saveOperation:` (page 995) Deprecated in Mac OS X v10.4
  Writes the receiver document's contents to a file. (Deprecated. Use `writeToURL:ofType:forSaveOperation:originalContentsURL:error:` (page 997) instead.)
- `writeToURL:ofType:` (page 996) Deprecated in Mac OS X v10.4
  Writes document data to a URL. (Deprecated. Use `writeToURL:ofType:error:` (page 996) instead.)
- `writeWithBackupToFile:ofType:saveOperation:` (page 998) Deprecated in Mac OS X v10.4
  This method is called by action methods to save document contents to a file. (Deprecated. Use `writeSafelyToURL:ofType:forSaveOperation:error:` (page 994) instead.)

Class Methods

canConcurrentlyReadDocumentsOfType:

Returns a Boolean value indicating whether the receiver reads multiple documents of the given type concurrently.

```
+ (BOOL)canConcurrentlyReadDocumentsOfType:(NSString *)typeName
```

Parameters

typeName
The string that identifies the document type.

Return Value

`NO` by default; subclasses can override to return `YES`, thereby causing documents of the specified type to be read concurrently.

Discussion

Your `NSDocument` subclass can implement this method to return `YES` to enable loading of documents concurrently, using background threads. When this facility is enabled in this way, `initWithContentsOfURL:ofType:error:` (page 957) executes on a background thread when opening files via the Open panel or from the Finder. This allows concurrent reading of multiple documents and also allows the application to be responsive while reading a large document.

The default implementation of this method returns `NO`. A subclass override should return `YES` only for document types whose reading is thread-safe, as described in "Multicore Considerations" (page 930). You should disable undo registration during document reading, which is a good idea even in the absence of concurrency.

If you are checking the current Apple Event for a search string, you should not enable concurrent document opening, because code handling a document opening triggered by an Apple Event cannot get the current Apple Event. This happens because the event is suspended until all documents are read to enable correct reporting of success or error.

Availability

Available in Mac OS X v10.6 and later.

See Also
+ readableTypes (page 939)

Declared In
NSDocument.h

isNativeType:

Returns a Boolean value indicating whether document data of the specified type is a native type—one the receiver can both read and write.

```
+ (BOOL)isNativeType:(NSString *)aType
```

Parameters

aType
The string that identifies the document type to test.

Return Value
YES if the document type is a native type; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ readableTypes (page 939)
+ writableTypes (page 940)

Declared In
NSDocument.h

readableTypes

Returns the types of data the receiver can read natively and any types filterable to that native type.

```
+ (NSArray *)readableTypes
```

Return Value
An array of NSString objects representing the readable document types.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ isNativeType: (page 939)
+ writableTypes (page 940)
+ canConcurrentlyReadDocumentsOfType: (page 938)

Related Sample Code
ImageApp
iSpend

Declared In
NSDocument.h

writableTypes

Returns the types of data the receiver can write natively and any types filterable to that native type.

```
+ (NSArray *)writableTypes
```

Return Value
An array of `NSString` objects representing the writable document types.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ isNativeType:` (page 939)
`+ readableTypes` (page 939)

Related Sample Code
FunHouse
ImageApp
iSpend

Declared In
`NSDocument.h`

Instance Methods

addWindowController:

Adds the specified window controller to the array of window controllers associated with the receiver.

```
- (void)addWindowController:(NSWindowController *)aController
```

Parameters
aController
The window controller that is added.

Discussion
An `NSDocument` object uses this list when it displays all document windows, sets window edited status upon an undo or redo operation, and modifies window titles. The method also sets the document outlet of the window controller to `self` if it is not already set. If you create window controllers by overriding `windowNibName` (page 992), this method is invoked automatically. If you create window controllers in `makeWindowControllers` (page 961) or in any other context, such as in response to a user event, you should invoke this method for each created window controller. To remove a window controller from the list of active controllers, send it the `NSWindowController` message `close` (page 3432).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setDocument:` (page 3437) (NSWindowController)

Related Sample Code
FunHouse
QTAudioContextInsert
QTKitFrameStepper
Reviews
Sketch+Accessibility

Declared In
`NSDocument.h`

autosavedContentsFileURL

Returns the location of the most recently autosaved document contents.

```
- (NSURL *)autosavedContentsFileURL
```

Return Value
The location of the most recently autosaved document contents.

Discussion
The default implementation of this method just returns whatever was stored by a previous invocation of the default implementation of `setAutosavedContentsFileURL:` (page 980).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAutosavedContentsFileURL:` (page 980)

Declared In
`NSDocument.h`

autosaveDocumentWithDelegate:didAutosaveSelector:contextInfo:

Autosaves the document's contents at an appropriate location.

```
- (void)autosaveDocumentWithDelegate:(id)delegate
    didAutosaveSelector:(SEL)didAutosaveSelector contextInfo:(void *)contextInfo
```

Parameters

delegate
The delegate to which the selector message is sent.

didAutosaveSelector
The selector of the message sent to the delegate.

contextInfo
Object passed with the callback to provide any additional context information.

Discussion
After autosaving, sends the message selected by *didAutosaveSelector* to the delegate, with *contextInfo* as the last argument. The method selected by *didAutosaveSelector* must have the same signature as:

```
- (void)document:(NSDocument *)document didAutosave:(BOOL)didAutosaveSuccessfully
  contextInfo:(void *)contextInfo
```

If an error occurs while autosaving, the method reports it to the user before sending the delegate a `succeeded:NO` message.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `autosavedContentsFileURL` (page 941)

Declared In
`NSDocument.h`

autosavingFileType

Returns the document type that should be used for an autosave operation.

```
- (NSString *)autosavingFileType
```

Return Value
The string that identifies the document type.

Discussion
The default implementation just returns `[self fileType]`. You can override this method and return `nil` in your override to completely disable autosaving of individual documents (because `NSDocumentController` does not send `autosaveDocumentWithDelegate:didAutosaveSelector:contextInfo:` to a document that has no autosaving file type). You can also override it if your application defines a document type that is specifically designed for autosaving, for example, one that efficiently represents document content changes instead of complete document contents.

Overriding `autosavingFileType` can result in incorrect behavior during reopening of autosaved documents. The `NSDocument` method `initForURL:withContentsOfURL:ofType:error:` (page 955), which is invoked during reopening of autosaved documents after a crash, takes two URLs, but only the type name of the autosaved contents file. The default implementation invokes `[self setFileType:]` with that type name, but that may not be the right thing to do if `autosavingFileType` returned something other than `fileType` (page 950) during document autosaving. If you override `autosavingFileType`, you probably need to override `initForURL:withContentsOfURL:ofType:error:` (page 955) too, and make the override invoke `setFileType:` (page 982) with the type of the actual document file, after invoking `super`. See TextEdit's `Document` class for an example of how to do this.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSDocument.h`

canCloseDocument

This method is no longer supported. (Available in Mac OS X v10.0 through Mac OS X v10.3. Use `canCloseDocumentWithDelegate:shouldCloseSelector:contextInfo:` (page 943) instead.)

```
- (BOOL)canCloseDocument
```

Availability
Available in Mac OS X v10.0 through Mac OS X v10.3.

Declared In
NSDocument.h

canCloseDocumentWithDelegate:shouldCloseSelector:contextInfo:

If the receiver is not dirty, this method immediately calls the *shouldCloseSelector* callback on the specified delegate with YES.

```
- (void)canCloseDocumentWithDelegate:(id)delegate
    shouldCloseSelector:(SEL)shouldCloseSelector contextInfo:(void *)contextInfo
```

Parameters

delegate
The delegate to which the selector message is sent.

shouldCloseSelector
The selector of the message sent to the delegate.

contextInfo
Object passed with the callback to provide any additional context information.

Discussion
If the receiver is dirty, an alert is presented giving the user a chance to save, not save, or cancel. If the user chooses to save, this method saves the document. If the save completes successfully, this method calls the callback with YES. If the save is canceled or otherwise unsuccessful, this method calls the callback with NO. This method may be called by `shouldCloseWindowController:delegate:shouldCloseSelector:contextInfo:` (page 986). It is also called by the NSDocumentController method `closeAllDocuments`. You should call it before you call `close` (page 943) if you are closing the document and want to give the user a chance to save any edits. Pass the *contextInfo* object with the callback.

The *shouldCloseSelector* callback method should have the following signature:

```
- (void)document:(NSDocument *)doc shouldClose:(BOOL)shouldClose
contextInfo:(void  *)contextInfo
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDocument.h

close

Closes all windows owned by the receiver and removes the receiver from the list of documents maintained by the document controller, which consequently releases it.

```
- (void)close
```

Discussion
This method closes the document immediately, without asking users if they want to save the document.

This method may not always be called. Additional information on application termination can be found in Graceful Application Termination.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `canCloseDocumentWithDelegate:shouldCloseSelector:contextInfo:` (page 943)
- `shouldCloseWindowController:delegate:shouldCloseSelector:contextInfo:` (page 986)

Related Sample Code
ThreadsExporter
ThreadsExportMovie
ThreadsImporter
ThreadsImportMovie
ZipBrowser

Declared In
`NSDocument.h`

dataOfType:error:

Creates and returns a data object that contains the contents of the document, formatted to a specified type.

```
- (NSData *)dataOfType:(NSString *)typeName error:(NSError **)outError
```

Parameters

typeName
The string that identifies the document type.

outError
On return, If the data object could not be created, a pointer to an error object that encapsulates the reason it could not be created.

Return Value
A data object containing the document contents, or, if the data object could not be created, `nil`.

Discussion
The default implementation of this method throws an exception because at least one of the writing methods (this method, `writeToURL:ofType:error:` (page 996), `fileWrapperOfType:error:` (page 951), or `writeToURL:ofType:forSaveOperation:originalContentsURL:error:` (page 997)) must be overridden.

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method instead invokes `dataRepresentationOfType:`*typeName* on `self` if `dataRepresentationOfType:` is overridden.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `writeToURL:ofType:error:` (page 996)

- fileWrapperOfType:error: (page 951)

Declared In
NSDocument.h

dataRepresentationOfType:

A primitive method overridden by subclasses to return a data object that represents the data of the receiver in a given type. (Deprecated in Mac OS X v10.4. Use dataOfType:error: (page 944) instead.)

```
- (NSData *)dataRepresentationOfType:(NSString *)aType
```

Discussion
A primitive method overridden by subclasses to return a data object that represents the data of the receiver in a given type (*aType*). The default implementation raises an NSInternalInconsistencyException. This method is invoked by the default implementation of fileWrapperRepresentationOfType:.

aType is the type name corresponding to the value of the CFBundleTypeName entry in the document type's Info.plist dictionary.

Here is a typical implementation:

```
//Document type name
NSString *MyDocumentType = @"Rich Text Format (RTF) document";

...

- (NSData *)dataRepresentationOfType:(NSString *)aType {
    NSAssert([aType isEqualToString:MyDocumentType], @"Unknown  type");
    return [textView RTFFromRange:NSMakeRange(0, [[textView textStorage]
length])];
}
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- loadDataRepresentation:ofType: (page 959)

Related Sample Code
FinalCutPro_AppleEvents

Declared In
NSDocument.h

displayName

Returns the name of the receiver as displayed in the title bars of the document's windows and in alert dialogs related to the document.

```
- (NSString *)displayName
```

Return Value
The display name of the receiver.

Discussion
If the document has been saved, the display name is the last component of the directory location of the saved file (for example, "`MyDocument`" if the path is "`/tmp/MyDocument.rtf`"). If the document is new, `NSDocument` makes the display name "Untitled *n*," where *n* is a number in a sequence of new and unsaved documents. The displayable name also takes into account whether the document's filename extension should be hidden. Subclasses of `NSWindowController` can override `windowTitleForDocumentDisplayName:` (page 3444) to modify the display name as it appears in window titles.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CoreTextArcCocoa
EnhancedAudioBurn
QTMetadataEditor

Declared In
`NSDocument.h`

fileAttributesToWriteToFile:ofType:saveOperation:

Returns the file attributes that should be written to the named document file of the specified type. (Deprecated in Mac OS X v10.4. Use `fileAttributesToWriteToURL:ofType:forSaveOperation:originalContentsURL:error:` (page 946) instead.)

```
- (NSDictionary *)fileAttributesToWriteToFile:(NSString *)fullDocumentPath
    ofType:(NSString *)docType saveOperation:(NSSaveOperationType)saveOperationType
```

Discussion
Returns the file attributes that should be written to the named document file of the specified type *docType*, as part of a particular *saveOperationType*. The set of valid file attributes is a subset of those understood by the `NSFileManager` class.

Availability
Available in Mac OS X v10.1 and later.
Deprecated in Mac OS X v10.4.

Declared In
`NSDocument.h`

fileAttributesToWriteToURL:ofType:forSaveOperation:originalContentsURL:error:

As a file is being saved, returns the attributes that should be written to a file or file package located by a URL, formatted to a specified type, for a particular kind of save operation.

```
- (NSDictionary *)fileAttributesToWriteToURL:(NSURL *)absoluteURL ofType:(NSString
    *)typeName forSaveOperation:(NSSaveOperationType)saveOperation
   originalContentsURL:(NSURL *)absoluteOriginalContentsURL error:(NSError
   **)outError
```

Parameters

absoluteURL

The location to which the document is being written.

typeName

The string that identifies the document type.

saveOperation

The type of save operation.

absoluteOriginalContentsURL

The location of the previously saved copy of the document (if not `nil`).

outError

On return, If the attributes could not be returned, a pointer to an error object that encapsulates the reason they could not be returned.

Return Value

A dictionary containing the attributes to be written, or `nil` if unsuccessful.

Discussion

The set of valid file attributes is a subset of those understood by the `NSFileManager` class. The default implementation of this method returns a dictionary with `NSFileHFSCreatorCode` and `NSFileHFSTypeCode` entries that have a value of 0 for `NSSaveOperation`, or a dictionary with an appropriate `NSFileExtensionHidden` entry for `NSSaveAsOperation` and `NSSaveToOperation`. You can override this method to customize the attributes that are written to document files.

This method is meant to be used just for attributes that need to be written for the first time, for `NSSaveAsOperation` and `NSSaveToOperation`.

Invokers of this method should silently ignore invalid attributes. Of particular interest is the `NSFileExtensionHidden` attribute, which is documented in `NSFileManager`.

Your subclass of `NSDocument` can override this method to control the attributes that are set during a save operation. An override of this method should return a copy of the dictionary returned by its superclass's version of this method, with appropriate alterations.

The dictionary returned by the default implementation of this method contains an `NSFileExtensionHidden` entry when that is appropriate. For Save As or Export commands (`NSSaveAsOperation` or `NSSaveToOperation`) the attributes dictionary contains an `NSFileExtensionHidden` entry whose value is the result of `[self fileNameExtensionWasHiddenInLastRunSavePanel]`. For autosaving (`NSAutosaveOperation`) the contents of documents that have already been saved (that is, not untitled documents), there's an `NSFileExtensionHidden` entry whose value is the same as that in the file attributes dictionary for the document itself, so the autosaved file has the same name extension hiding as the real document.

An override of `writeSafelyToURL:ofType:forSaveOperation:error:` (page 994) should invoke this method and set the returned attributes on the written document file, possibly using the `NSFileManager` method `changeFileAttributes:atPath:`.

Implementers of overrides of this method should not assume that:

- The file pointed to by *absoluteURL* at the moment the method is invoked, if there is one, is related to the document itself. It may be an unrelated file that is about to be overwritten.
- The `fileURL` (page 951) or `fileType` (page 950) method will return anything useful at the moment.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSDocument.h`

fileModificationDate

Returns the last known modification date of the document's on-disk representation.

```
- (NSDate *)fileModificationDate
```

Return Value
The file modification date.

Discussion
The `NSDocument` default file saving machinery uses this information to warn the user when the on-disk representation of an open document has been modified by something other than the current application.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setFileModificationDate:` (page 981)

Declared In
`NSDocument.h`

fileName

Returns the filename (as a fully qualified path) under which the receiver has been saved. (Deprecated in Mac OS X v10.4. Use `fileURL` (page 951) instead.)

```
- (NSString *)fileName
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
`- setFileName:` (page 981)

Related Sample Code
QTAudioExtractionPanel
QTKitPlayer

Declared In
`NSDocument.h`

fileNameExtensionForType:saveOperation:

Returns a filename extension that can be appended to a base filename, for a specified file type and kind of save operation.

```
- (NSString *)fileNameExtensionForType:(NSString *)typeName
    saveOperation:(NSSaveOperationType)saveOperation
```

Parameters

typeName
The file type.

saveOperation
The kind of save operation.

Return Value
The filename extension.

Discussion
The default implementation of this method invokes `preferredFileNameExtensionForType:` on the shared workspace object if the type is a UTI or, if it is not, for backward binary compatibility with Mac OS X v10.4 and earlier, invokes `fileExtensionsFromType:` (page 1015) on the shared document controller and chooses the first filename extension in the returned array.

You can override this method to customize the appending of extensions to filenames by `NSDocument`. In Mac OS X v10.5, it's only invoked from two places in the Application Kit:

1. The `autosaveDocumentWithDelegate:didAutosaveSelector:contextInfo:` (page 941) method uses this method when creating a new filename for the autosaved contents.

2. The `handleSaveScriptCommand:` (page 953) method uses this method when adding an extension to the filename specified by a script.

In all other cases, the name of any file being saved will have been fully specified by the user with the Save panel (whether they know it or not).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSDocument.h`

fileNameExtensionWasHiddenInLastRunSavePanel

Returns `YES` if a Save panel was presented by this document and the user chose to hide the name extension of the file that was selected in that Save panel.

```
- (BOOL)fileNameExtensionWasHiddenInLastRunSavePanel
```

Return Value
`YES` if a Save panel was presented and the user chose to hide the extension; otherwise, `NO`.

Availability
Available in Mac OS X v10.1 and later.

Declared In
NSDocument.h

fileNameFromRunningSavePanelForSaveOperation:

Returns the filename entered into the Save panel. (Available in Mac OS X v10.0 through Mac OS X v10.3. Use `saveDocumentWithDelegate:didSaveSelector:contextInfo:` (page 978) instead.)

```
- (NSString
    *)fileNameFromRunningSavePanelForSaveOperation:(NSSaveOperationType)saveOperation
```

Availability
Available in Mac OS X v10.0 through Mac OS X v10.3.

Declared In
NSDocument.h

fileType

Returns the document type under which the receiver is saved.

```
- (NSString *)fileType
```

Return Value
The string that identifies the document type.

Discussion
When a document is saved, the type is determined by the entries in the application's information property list (specified in `Info.plist`).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setFileType:` (page 982)

Related Sample Code
ImageApp
ThreadsExporter
ThreadsImporter
ThreadsImportMovie

Declared In
NSDocument.h

fileTypeFromLastRunSavePanel

Returns the file type that was last selected in the Save panel.

```
- (NSString *)fileTypeFromLastRunSavePanel
```

Return Value
The string that identifies the document type.

Discussion
This type is primarily used by the `saveDocument:` (page 976), `saveDocumentAs:` (page 977), and `saveDocumentTo:` (page 977) methods to determine the type the user chose after the Save panel has been run.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ImageApp

Declared In
`NSDocument.h`

fileURL

Returns the location of the document's on-disk representation.

```
- (NSURL *)fileURL
```

Return Value
The document's location.

Discussion
The default implementation of this method returns whatever was stored by a previous invocation of the default implementation of `setFileURL:` (page 982). For backward binary compatibility with Mac OS X v10.3 and earlier, if `fileName` (page 948) is overridden, the default implementation of this method instead invokes `[self fileName]` and returns the result as a URL.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setFileURL:` (page 982)

Related Sample Code
FunHouse
ImageApp
iSpend
QTMetadataEditor
QuickLookDownloader

Declared In
`NSDocument.h`

fileWrapperOfType:error:

Creates and returns a file wrapper that contains the contents of the document, formatted to the specified type.

```
- (NSFileWrapper *)fileWrapperOfType:(NSString *)typeName error:(NSError **)outError
```

Parameters

typeName

The string that identifies the document type.

outError

On return, If the file wrapper could not be created, a pointer to an error object that encapsulates the reason it could not be created.

Return Value

A file wrapper containing the document contents, or, if the file wrapper could not be created, `nil`.

Discussion

For backward binary compatibility with Mac OS X v10.3 and earlier, if `fileWrapperRepresentationOfType:` (page 952) is overridden, the default implementation of this method instead invokes `[self fileWrapperRepresentationOfType:typeName]`.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- dataOfType:error:` (page 944)

Declared In

`NSDocument.h`

fileWrapperRepresentationOfType:

Returns an NSFileWrapper object that represents the data of the receiver in a given type. (Deprecated in Mac OS X v10.4. Use `fileWrapperOfType:error:` (page 951) instead.)

```
- (NSFileWrapper *)fileWrapperRepresentationOfType:(NSString *)aType
```

Discussion

Returns an `NSFileWrapper` object that represents the data of the receiver in a given type (*aType*). This method invokes `dataRepresentationOfType:` to get the data object from which to create a plain-file file wrapper. Subclasses can override this method if `dataRepresentationOfType:` is not adequate for their needs. This method is invoked by the default implementation of `writeToFile:ofType:` (page 995).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

See Also

`- loadFileWrapperRepresentation:ofType:` (page 960)

Declared In

`NSDocument.h`

handleCloseScriptCommand:

Handles the Close AppleScript command by attempting to close the document.

```
- (id)handleCloseScriptCommand:(NSCloseCommand *)command
```

Parameters

command

A Close AppleScript command object.

Discussion

Extracts Close command arguments from the *command* object and uses them to determine how to close the document—specifically, whether to ignore unsaved changes, save changes automatically, or ask the user and to identify the file in which to save the document (by default, the file that was opened or previously saved to). A Close AppleScript command may specify more than one document to close. If so, a message is sent to each document object.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDocumentScripting.h

handlePrintScriptCommand:

Handles the Print AppleScript command by attempting to print the document.

```
- (id)handlePrintScriptCommand:(NSScriptCommand *)command
```

Parameters

command

An AppleScript command object.

Discussion

Extracts Print command arguments from the *command* object and uses them to determine how to print the document—specifically, any print settings and whether to show the Print dialog. A Print AppleScript command may specify more than one document to print. If so, a message is sent to each document.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDocumentScripting.h

handleSaveScriptCommand:

Handles the Save AppleScript command by attempting to save the document.

```
- (id)handleSaveScriptCommand:(NSScriptCommand *)command
```

Parameters

command

An AppleScript command object.

Discussion

Extracts Save command arguments from the *command* object and uses them to determine the file in which to save the document and the file type.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDocumentScripting.h

hasUnautosavedChanges

Return YES if the document has changes that have not been autosaved, as determined by the history of previous invocations of updateChangeCount: (page 988).

```
- (BOOL)hasUnautosavedChanges
```

Return Value
YES if the document has changes that have not been autosaved; otherwise, NO.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSDocument.h

hasUndoManager

Returns a Boolean value indicating whether the receiver owns or should own an NSUndoManager object.

```
- (BOOL)hasUndoManager
```

Return Value
YES if the receiver has its own undo manager; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setHasUndoManager: (page 983)

Declared In
NSDocument.h

init

Initializes and returns an empty NSDocument object.

```
- (id)init
```

Return Value
An initialized NSDocument object.

Discussion
This initializer (the designated initializer) is invoked by each of the other NSDocument initialization methods.

You can override this method to perform initialization that must be done both when creating new empty documents and when opening existing documents. Your override must invoke `super` to initialize private `NSDocument` instance variables. It must never return `nil`. If an error can occur during object initialization, check for the error in an override of `initWithType:error:` (page 957), `initWithContentsOfURL:ofType:error:` (page 957), or `initForURL:withContentsOfURL:ofType:error:` (page 955), because those methods can return `NSError` objects.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTAudioContextInsert
QTMetadataEditor
Simple Bindings Adoption
Sketch-112
ZipBrowser

Declared In
`NSDocument.h`

initForURL:withContentsOfURL:ofType:error:

Initializes a document located by a URL of a specified type, but by reading the contents for the document from a different URL.

```
- (id)initForURL:(NSURL *)absoluteDocumentURL withContentsOfURL:(NSURL
    *)absoluteDocumentContentsURL ofType:(NSString *)typeName error:(NSError
    **)outError
```

Parameters

absoluteDocumentURL
The URL where the document is located.

absoluteDocumentContentsURL
The URL from which the contents of the document are obtained.

typeName
The string that identifies the document type.

outError
On return, If initialization is unsuccessful, a pointer to an error object that encapsulates the reason the document could not be created.

Return Value
The initialized `NSDocument` object, or, if the document could not be created, `nil`.

Discussion
The *absoluteDocumentURL* argument is `nil` if the initializing is part of the reopening of an autosaved document when the autosaved document was never explicitly saved.

During reopening of autosaved documents, this method uses the following `NSDocumentChangeType` constant to indicate that an autosaved document is being reopened:

```
NSChangeReadOtherContents
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSDocument.h

initWithContentsOfFile:ofType:

Initializes and returns an NSDocument object. (Deprecated in Mac OS X v10.4. Use initWithContentsOfURL:ofType:error: (page 957) instead.)

```
- (id)initWithContentsOfFile:(NSString *)fileName ofType:(NSString *)docType
```

Discussion
Initializes and returns an NSDocument object of document type *docType* containing data stored in the file *fileName*. In opening the file, invokes the readFromFile:ofType: (page 968) method. If the document successfully opens the file, it calls setFileName: and setFileType: (page 982) with *fileName* and *docType*, respectively, as arguments. If the file cannot be opened, or the document is unable to load the contents of the file, this method returns nil. This initializer is typically invoked by the NSDocumentController method makeDocumentWithContentsOfFile:ofType: (page 1017).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitAdvancedDocument
QTKitImport
QTKitPlayer

Declared In
NSDocument.h

initWithContentsOfURL:ofType:

Initializes and returns an NSDocument object of a given document type. (Deprecated in Mac OS X v10.4. Use initWithContentsOfURL:ofType:error: (page 957) instead.)

```
- (id)initWithContentsOfURL:(NSURL *)aURL ofType:(NSString *)docType
```

Discussion
Initializes and returns an NSDocument object of document type *docType* containing data stored at *aURL*. In opening the location, invokes the readFromURL:ofType: (page 969) method. If the document successfully opens the location, it calls setFileName: and setFileType: (page 982) with the location's path and *docType*, respectively, as arguments. If the location cannot be opened, or the document is unable to load the contents of the location, this method returns nil. This initializer is typically invoked by the NSDocumentController method makeDocumentWithContentsOfURL:ofType: (page 1018).

Availability
Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In
NSDocument.h

initWithContentsOfURL:ofType:error:

Initializes a document located by a URL of a specified type.

```
- (id)initWithContentsOfURL:(NSURL *)absoluteURL ofType:(NSString *)typeName
    error:(NSError **)outError
```

Parameters

absoluteURL
The URL from which the contents of the document are obtained.

typeName
The string that identifies the document type.

outError
On return, If initialization is unsuccessful, a pointer to an error object that encapsulates the reason the document could not be created.

Return Value
The initialized NSDocument object, or, if the document could not be created, nil.

Discussion
You can override this method to customize the reopening of autosaved documents.

This method is invoked by the NSDocumentController method makeDocumentWithContentsOfURL:ofType:error: (page 1019). The default implementation of this method invokes init (page 954), readFromURL:ofType:error: (page 970), setFileURL: (page 982), setFileType: (page 982), and setFileModificationDate: (page 981).

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method instead invokes initWithContentsOfFile:ofType: (page 956) if it is overridden and the URL uses the file: scheme. It still invokes setFileModificationDate: in this situation.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
QTKitCreateMovie
QTKitFrameStepper
QTMetadataEditor

Declared In
NSDocument.h

initWithType:error:

Initializes a document of a specified type.

```
- (id)initWithType:(NSString *)typeName error:(NSError **)outError
```

Parameters

typeName

The string that identifies the document type.

outError

On return, If initialization is unsuccessful, a pointer to an error object that encapsulates the reason the document could not be created.

Return Value
The initialized `NSDocument` object, or, if the document could not be created, `nil`.

Discussion
The default implementation of this method just invokes `[self init]` and `[self setFileType:typeName]`.

You can override this method to perform initialization that must be done when creating new documents but should not be done when opening existing documents. Your override should typically invoke `super`, or at least it must invoke `init` (page 954), the `NSDocument` designated initializer, to initialize the `NSDocument` private instance variables.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSDocument.h`

isDocumentEdited

Returns `YES` if the receiver has changes that have not been saved, `NO` otherwise.

```
- (BOOL)isDocumentEdited
```

Return Value
`YES` if the receiver has been edited; otherwise, `NO`.

Discussion
The edited status of each document window reflects the document's edited status.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- updateChangeCount:` (page 988)
`- setDocumentEdited:` (page 3379) (NSWindow)

Related Sample Code
FunHouse

Declared In
`NSDocument.h`

keepBackupFile

Returns a Boolean value indicating whether the receiver should keep the backup files created before document data is written to a file (`NO` by default).

```
- (BOOL)keepBackupFile
```

Return Value
NO by default; subclasses can override to return YES, thereby causing backup files to be kept.

Availability
Available in Mac OS X v10.0 and later.

See Also
- writeToFile:ofType: (page 995)

Declared In
NSDocument.h

lastComponentOfFileName

Returns the document name in terms of the scripting name property (the name a script writer would use to specify the document in a script).

```
- (NSString *)lastComponentOfFileName
```

Return Value
The scripting name of the document.

Discussion
Note that this name may be different than the name returned by fileName (page 948) or used in methods such as writeToFile:ofType: (page 995).

Availability
Available in Mac OS X v10.0 and later.

See Also
- displayName (page 945)

Declared In
NSDocumentScripting.h

loadDataRepresentation:ofType:

Overridden by subclasses to load document data. (Deprecated in Mac OS X v10.4. Use readFromData:ofType:error: (page 967) instead.)

```
- (BOOL)loadDataRepresentation:(NSData *)docData ofType:(NSString *)docType
```

Discussion
Overridden by subclasses to load document data (*docData*) of type *docType* into the receiver, display it in windows, and return whether the operation was successful. This method is typically invoked by loadFileWrapperRepresentation:ofType: (page 960) after an NSData object is created from the contents of the file wrapper (which can include directories). The default implementation raises an NSInternalInconsistencyException. Subclasses must override this method unless they override readFromFile:ofType: (page 968) or loadFileWrapperRepresentation:ofType: (page 960) to do specialized reading and loading of document data.

The *docType* argument is the type name corresponding to the value of the `CFBundleTypeName` entry in the document type's `Info.plist` dictionary.

Here is an example implementation:

```
//Document type name
NSString *MyDocumentType = @"Rich Text Format (RTF) document";

...

- (BOOL)loadDataRepresentation:(NSData *)data ofType:(NSString *)aType  {
    NSAssert([aType isEqualToString: MyDocumentType], @"Unknown  type");
    fileContents = [data copyWithZone:[self zone]];
    return YES;
}
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
`- dataRepresentationOfType:` (page 945)

Related Sample Code
FinalCutPro_AppleEvents

Declared In
`NSDocument.h`

loadFileWrapperRepresentation:ofType:

Loads document data from a given file wrapper. (Deprecated in Mac OS X v10.4. Use `readFromFileWrapper:ofType:error:` (page 969) instead.)

```
- (BOOL)loadFileWrapperRepresentation:(NSFileWrapper *)wrapper ofType:(NSString
    *)docType
```

Discussion
Loads document data in file wrapper *wrapper* of type *docType* into the receiver, displays it in windows, and returns whether the operation was successful. If *wrapper* is a simple file, it invokes `loadDataRepresentation:ofType:` (page 959) to load the data. If *wrapper* is a directory, it returns `NO` by default; subclasses can override to handle file wrappers that are directories. This method is typically invoked by `readFromFile:ofType:` (page 968) after it creates an NSData object from the contents of the file.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
`- fileWrapperRepresentationOfType:` (page 952)

Declared In
`NSDocument.h`

makeWindowControllers

Subclasses may override this method to create the initial window controller(s) for the document.

```
- (void)makeWindowControllers
```

Discussion
The base class implementation creates an `NSWindowController` object with `windowNibName` (page 992) and with the document as the file's owner if `windowNibName` (page 992) returns a name. If you override this method to create your own window controllers, be sure to use `addWindowController:` (page 940) to add them to the document after creating them.

This method is called by the `NSDocumentController` `open...` methods, but you might want to call it directly in some circumstances.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- windowControllers` (page 991)

Related Sample Code
QTAudioExtractionPanel
QTKitImport
QTKitPlayer
QTMetadataEditor
Sketch-112

Declared In
`NSDocument.h`

objectSpecifier

Returns an object specifier for the document.

```
- (NSScriptObjectSpecifier *)objectSpecifier
```

Return Value
The document object specifier.

Discussion
An object specifier represents an AppleScript reference form, which is a natural-language expression such as `words 10 through 20` or `front document`. During script processing, an object contained by a document (such as the `second paragraph` or the `third rectangle`) may need to specify its container (the document).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QuickLookSketch
Sketch+Accessibility

Declared In
`NSDocumentScripting.h`

preparePageLayout:

Invoked by `runModalPageLayoutWithPrintInfo:` (page 973) and `runModalPageLayoutWithPrintInfo:delegate:didRunSelector:contextInfo:` (page 973) to do any customization of the Page Layout panel *pageLayout*, such as adding an accessory view.

```
- (BOOL)preparePageLayout:(NSPageLayout *)pageLayout
```

Parameters

pageLayout

The page layout panel to prepare.

Return Value

`YES` if successfully prepared; otherwise, `NO`.

Discussion

The default implementation is empty and returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDocument.h`

prepareSavePanel:

Invoked by `runModalSavePanelForSaveOperation:delegate:didSaveSelector:contextInfo:` (page 975) to do any customization of the given Save panel.

```
- (BOOL)prepareSavePanel:(NSSavePanel *)savePanel
```

Parameters

savePanel

The Save panel.

Return Value

`YES` if successfully prepared; otherwise, `NO`.

Discussion

The default implementation is empty and returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDocument.h`

presentError:

Presents an error alert to the user as a modal panel.

```
- (BOOL)presentError:(NSError *)error
```

Parameters

error

The error object encapsulating the information to present to the user.

Return Value

`YES` if error recovery was done; otherwise, `NO`.

Discussion

This method does not return until the user dismisses the alert and, if the error has recovery options and a recovery delegate, the error's recovery delegate is sent an `attemptRecoveryFromError:optionIndex:` message.

The `NSDocument` default implementation of this method is equivalent to that of `NSResponder` and treats the shared `NSDocumentController` as the next responder and forwards these messages to it.

The default implementation of this method invokes `willPresentError:` (page 989) to give subclasses an opportunity to customize error presentation. You should not override this method but should instead override `willPresentError:` (page 989).

Availability

Available in Mac OS X v10.4 and later.

See Also

- `willPresentError:` (page 989)
- `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 963)

Declared In

`NSDocument.h`

presentError:modalForWindow:delegate:didPresentSelector:contextInfo:

Presents an error alert to the user as a modal panel.

```
- (void)presentError:(NSError *)error modalForWindow:(NSWindow *)window
    delegate:(id)delegate didPresentSelector:(SEL)didPresentSelector
    contextInfo:(void *)contextInfo
```

Parameters

error

The error object encapsulating the information to present to the user.

window

The window to which the modal alert belongs.

delegate

The delegate to which the selector message is sent.

didPresentSelector

The selector of the message sent to the delegate.

contextInfo

Object passed with the callback to provide any additional context information.

Discussion

When the user dismisses the alert and any recovery possible for the error and chosen by the user has been attempted, sends the message *didPresentSelector* to the specified *delegate*.

The NSDocument default implementation of this method is equivalent to that of NSResponder and treats the shared NSDocumentController object as the next responder and forwards these messages to it. The default implementations of several NSDocument methods invoke this method.

The default implementation of this method invokes willPresentError: (page 989) to give subclasses an opportunity to customize error presentation. You should not override this method but should instead override willPresentError:.

The method selected by *didPresentSelector* must have the same signature as:

```
- (void)didPresentErrorWithRecovery:(BOOL)didRecover contextInfo:(void
*)contextInfo
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- presentError: (page 962)
- willPresentError: (page 989)

Related Sample Code
ZipBrowser

Declared In
NSDocument.h

printDocument:

Prints the receiver in response to the user choosing the Print menu command.

```
- (void)printDocument:(id)sender
```

Parameters

sender
The control sending the message.

Discussion
An NSDocument object receives this action message as it travels up the responder chain. The default implementation invokes printDocumentWithSettings:showPrintPanel:delegate:didPrintSelector:contextInfo: (page 965).

Availability
Available in Mac OS X v10.0 and later.

See Also
- printInfo (page 966)
- runPageLayout: (page 975)
- setPrintInfo: (page 984)
- shouldChangePrintInfo: (page 985)

Declared In
NSDocument.h

printDocumentWithSettings:showPrintPanel:delegate:didPrintSelector:contextInfo:

Prints the document.

```
- (void)printDocumentWithSettings:(NSDictionary *)printSettings
    showPrintPanel:(BOOL)showPrintPanel delegate:(id)delegate
    didPrintSelector:(SEL)didPrintSelector contextInfo:(void *)contextInfo
```

Parameters

printSettings

The print settings dictionary to use.

showPrintPanel

A Boolean value indicating whether the print panel is shown.

delegate

The delegate to which the selector message is sent.

didPrintSelector

The selector of the message sent to the delegate.

contextInfo

Object passed with the callback to provide any additional context information.

Discussion

If showing of the print panel is specified by *showPrintPanel*, the method presents it first and prints only if the user approves the panel. The `NSPrintInfo` attributes in the passed-in *printSettings* dictionary are added to a copy of the document's print info, and the resulting print info settings are used for the operation. When printing is complete or canceled, the method sends the message selected by *didPrintSelector* to the *delegate*, with the *contextInfo* as the last argument. The method selected by *didPrintSelector* must have the same signature as:

```
- (void)document:(NSDocument *)document didPrint:(BOOL)didPrintSuccessfully
contextInfo: (void *)contextInfo
```

The default implementation of this method invokes `printOperationWithSettings:error:` (page 966). If `nil` is returned it presents the error to the user in a document-modal panel before messaging the delegate. Otherwise it invokes `[thePrintOperation setShowsPrintPanel:showPrintPanel]` then `[self runModalPrintOperation:thePrintOperation delegate:delegate didRunSelector:didPrintSelector contextInfo:contextInfo]`.

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method invokes `printShowingPrintPanel:` (page 967) if it is overridden. When doing this it uses private functionality to arrange for the print settings to take effect (despite the fact that the override of `printShowingPrintPanel:` can't possibly know about them) and to get notified when the print operation has been completed, so it can message the delegate at the correct time. Correct messaging of the delegate is necessary for correct handling of the Print Apple event.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `printOperationWithSettings:error:` (page 966)

Declared In

`NSDocument.h`

printInfo

Returns the receiver's customized `NSPrintInfo` object or the default `NSPrintInfo` instance.

```
- (NSPrintInfo *)printInfo
```

Return Value
The receiver's `NSPrintInfo` object.

Discussion
The document's copy of the `NSPrintInfo` object can either be directly set or set as a result of running the Page Layout panel. A subclass can override this method to always return the shared `NSPrintInfo` instance if it does not want its own copy.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `runPageLayout:` (page 975)
- `setPrintInfo:` (page 984)
- `shouldChangePrintInfo:` (page 985)

Related Sample Code
ImageApp
QTAudioContextInsert
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSDocument.h`

printOperationWithSettings:error:

Creates a print operation and returns it if successful.

```
- (NSPrintOperation *)printOperationWithSettings:(NSDictionary *)printSettings
    error:(NSError **)outError
```

Parameters
printSettings
The print settings dictionary to use.

outError
On return, If the print operation could not be created, a pointer to an error object that encapsulates the reason it could not be created.

Return Value
The print operation, or `nil` if unsuccessful.

Discussion
The print operation can be run to print the document's current contents. The `NSPrintInfo` attributes in the passed-in *`printSettings`* dictionary are added to a copy of the document's print info, and the resulting print info is used for the operation. The default implementation of this method does nothing. You must override it to enable printing in your application.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- printDocumentWithSettings:showPrintPanel:delegate:didPrintSelector:contextInfo:` (page 965)

Declared In
`NSDocument.h`

printShowingPrintPanel:

Overridden by subclasses to print the current document's (the receiver's) data. (Deprecated in Mac OS X v10.4. Use `printDocumentWithSettings:showPrintPanel:delegate:didPrintSelector:contextInfo:` (page 965) instead.)

```
- (void)printShowingPrintPanel:(BOOL)flag
```

Discussion
Overridden by subclasses to print the current document's (the receiver's) data; if *`flag`* is `YES`, the implementation should first display the Print panel. This method is typically invoked by `printDocument:` with an argument of `YES`. The default implementation does nothing. If there is any printing information other than that encoded in the receiver's `NSPrintInfo` object, subclasses should get it here.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
`- printInfo` (page 966)

Declared In
`NSDocument.h`

readFromData:ofType:error:

Sets the contents of this document by reading from data of a specified type and returns `YES` if successful.

```
- (BOOL)readFromData:(NSData *)data ofType:(NSString *)typeName error:(NSError
    **)outError
```

Parameters

`data`

The data object from which the document contents are read.

typeName
The string that identifies the document type.

outError
On return, If the document contents could not be read, a pointer to an error object that encapsulates the reason they could not be read.

Return Value
`YES` if the document contents could be read; otherwise, `NO`.

Discussion
The default implementation of this method throws an exception because at least one of the three reading methods (this method, `readFromURL:ofType:error:` (page 970), `readFromFileWrapper:ofType:error:` (page 969)), or every method that may invoke `readFromURL:ofType:error:` (page 970), must be overridden.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSDocument.h`

readFromFile:ofType:

Reads and loads document data of the given type from the given file. (Deprecated in Mac OS X v10.4. Use `readFromURL:ofType:error:` (page 970) instead.)

```
- (BOOL)readFromFile:(NSString *)fileName ofType:(NSString *)docType
```

Discussion
Reads and loads document data of type *docType* from the file *fileName*, returning whether the operation was successful. This method invokes `loadDataRepresentation:ofType:` and is invoked when the receiver is first created and initialized by `initWithContentsOfFile:ofType:`. It uses `NSData initWithContentsOfFile:` to get the document data.

This method is one of the location-based primitives. Subclasses can override this method instead of overriding `loadDataRepresentation:ofType:` to read and load document data. Subclasses that handle file packages such as RTFD or that treat locations of files as anything other than paths should override this method. Override implementations of this method can filter the document data using `NSPasteboard` or other filtering services.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
`- dataRepresentationOfType:` (page 945)
`- writeToFile:ofType:` (page 995)

Declared In
`NSDocument.h`

readFromFileWrapper:ofType:error:

Sets the contents of this document by reading from a file wrapper of a specified type.

```
- (BOOL)readFromFileWrapper:(NSFileWrapper *)fileWrapper ofType:(NSString *)typeName
    error:(NSError **)outError
```

Parameters

fileWrapper

The file wrapper from which the document contents are read.

typeName

The string that identifies the document type.

outError

On return, If the document contents could not be read, a pointer to an error object that encapsulates the reason they could not be read.

Return Value

`YES` if the document contents could be read; otherwise, `NO`.

Discussion

The default implementation of this method invokes `[self readFromData:[fileWrapper regularFileContents] ofType:typeName error:outError]`.

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method instead invokes `[self loadFileWrapperRepresentation:fileWrapper ofType:typeName]` if `loadFileWrapperRepresentation:ofType:` (page 960) is overridden.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `readFromURL:ofType:error:` (page 970)
- `readFromData:ofType:error:` (page 967)

Declared In

`NSDocument.h`

readFromURL:ofType:

Reads and loads document data. (Deprecated in Mac OS X v10.4. Use `readFromURL:ofType:error:` (page 970) instead.)

```
- (BOOL)readFromURL:(NSURL *)aURL ofType:(NSString *)docType
```

Discussion

Reads and loads document data of type *docType* from the URL *aURL*, returning whether the operation was successful. This method only supports URLs of the `file:` scheme and calls `readFromFile:ofType:` (page 968).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In
NSDocument.h

readFromURL:ofType:error:

Sets the contents of this document by reading from a file or file package, of a specified type, located by a URL.

```
- (BOOL)readFromURL:(NSURL *)absoluteURL ofType:(NSString *)typeName error:(NSError **)outError
```

Parameters

absoluteURL
The location from which the document contents are read.

typeName
The string that identifies the document type.

outError
On return, If the document contents could not be read, a pointer to an error object that encapsulates the reason they could not be read.

Return Value
YES if the document contents could be read; otherwise, NO.

Discussion
The default implementation of this method just creates an NSFileWrapper and invokes `[self readFromFileWrapper:theFileWrapper ofType:typeName error:outError]`.

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method instead invokes `[self readFromFile:[absoluteURL path] ofType:typeName]` if `readFromFile:ofType:` (page 968) is overridden and the URL uses the `file:` scheme.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `readFromFileWrapper:ofType:error:` (page 969)
- `readFromData:ofType:error:` (page 967)

Declared In
NSDocument.h

removeWindowController:

Removes the specified window controller from the receiver's array of window controllers.

```
- (void)removeWindowController:(NSWindowController *)windowController
```

Parameters

windowController
The window controller that is removed.

Discussion
A document with no window controllers is not necessarily closed. However, a window controller can be set to close its associated document when the window is closed or the window controller is deallocated.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `shouldCloseDocument` (page 3440) (`NSWindowController`)

Declared In
`NSDocument.h`

revertDocumentToSaved:

The action of the File menu item Revert in a document-based application.

```
- (void)revertDocumentToSaved:(id)sender
```

Parameters

sender
The control sending the message.

Discussion
The default implementation of this method presents an alert dialog giving the user the opportunity to cancel the operation. If the user chooses to continue, the method ensures that any editor registered using the Cocoa Bindings `NSEditorRegistration` informal protocol has discarded its changes and then invokes `revertToContentsOfURL:ofType:error:` (page 971). If that returns `NO`, the method presents the error to the user in an document-modal alert dialog.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `updateChangeCount:` (page 988)

Declared In
`NSDocument.h`

revertToContentsOfURL:ofType:error:

Discards all unsaved document modifications and replaces the document's contents by reading a file or file package located by a URL of a specified type.

```
- (BOOL)revertToContentsOfURL:(NSURL *)absoluteURL ofType:(NSString *)typeName
    error:(NSError **)outError
```

Parameters

absoluteURL
The location from which the document contents are read.

typeName
The string that identifies the document type.

outError
On return, If the document could not be reverted, a pointer to an error object that encapsulates the reason it could not be reverted.

Return Value
YES if the document could be reverted; otherwise, NO.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSDocument.h

revertToSavedFromFile:ofType:

Reverts the receiver to the data stored in the file system. (Deprecated in Mac OS X v10.4. Use revertToContentsOfURL:ofType:error: (page 971) instead.)

```
- (BOOL)revertToSavedFromFile:(NSString *)fileName ofType:(NSString *)type
```

Discussion
Reverts the receiver to the data stored in the file system in file named *fileName* of file type *type*. Invokes readFromFile:ofType: (page 968) and returns whether that method successfully read the file and processed the document data.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- revertDocumentToSaved: (page 971)

Declared In
NSDocument.h

revertToSavedFromURL:ofType:

Reverts the receiver. (Deprecated in Mac OS X v10.4. Use revertToContentsOfURL:ofType:error: (page 971) instead.)

```
- (BOOL)revertToSavedFromURL:(NSURL *)aURL ofType:(NSString *)type
```

Discussion
Reverts the receiver to the data stored at *aURL* of type *type*. Invokes readFromURL:ofType: (page 969) and returns whether that method successfully read the file and processed the document data.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- revertDocumentToSaved: (page 971)

Declared In
NSDocument.h

runModalPageLayoutWithPrintInfo:

Runs the page layout modal panel with the receiver's printing information object. (Deprecated in Mac OS X v10.4. Use runModalPageLayoutWithPrintInfo:delegate:didRunSelector:contextInfo: (page 973) instead.)

```
- (NSInteger)runModalPageLayoutWithPrintInfo:(NSPrintInfo *)printInfo
```

Discussion
Runs the page layout modal panel with the receiver's printing information object (*printInfo*) as argument and returns the result constant (indicating the button pressed by the user). To run as sheet on the receiver's document window, use runModalPageLayoutWithPrintInfo:delegate:didRunSelector:contextInfo: (page 973) instead.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- shouldChangePrintInfo: (page 985)
- runModalWithPrintInfo: (page 1857) (NSPageLayout)

Declared In
NSDocument.h

runModalPageLayoutWithPrintInfo:delegate:didRunSelector:contextInfo:

Runs the modal page layout panel with the receiver's printing information object

```
- (void)runModalPageLayoutWithPrintInfo:(NSPrintInfo *)printInfo
    delegate:(id)delegate didRunSelector:(SEL)didRunSelector contextInfo:(void
    *)contextInfo
```

Parameters
printInfo
The NSPrintInfo object for the page layout panel to use.
delegate
The delegate to which the selector message is sent.
didRunSelector
The selector of the message sent to the delegate.
contextInfo
Object passed with the callback to provide any additional context information.

Discussion
Invoked from the action method runPageLayout: (page 975). Presents the page layout panel application modally if there is no document window to which it can be presented document modally.

When the panel is dismissed, *delegate* is sent a *didRunSelector* message. The *didRunSelector* callback method should have the following signature:

```
- (void)documentDidRunModalPageLayout:(NSDocument *)document
accepted:(BOOL)accepted  contextInfo:(void *)contextInfo
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDocument.h

runModalPrintOperation:delegate:didRunSelector:contextInfo:

Runs the specified print operation modally.

```
- (void)runModalPrintOperation:(NSPrintOperation *)printOperation
    delegate:(id)delegate didRunSelector:(SEL)didRunSelector contextInfo:(void
    *)contextInfo
```

Parameters

printOperation
The print operation to run.

delegate
The delegate to which the selector message is sent.

didRunSelector
The selector of the message sent to the delegate.

contextInfo
Object passed with the callback to provide any additional context information.

Discussion
Overrides of `printShowingPrintPanel:` (page 967) can invoke this method.

When the panel is dismissed, *delegate* is sent a *didRunSelector* message. Pass the *contextInfo* object with the callback. The *didRunSelector* callback method should have the following signature:

```
- (void)documentDidRunModalPrintOperation:(NSDocument *)document
success:(BOOL)success  contextInfo:(void *)contextInfo
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer

Declared In
NSDocument.h

runModalSavePanelForSaveOperation:delegate:didSaveSelector:contextInfo:

Presents a modal Save panel to the user, then tries to save the document if the user approves the panel.

```
- (void)runModalSavePanelForSaveOperation:(NSSaveOperationType)saveOperation
    delegate:(id)delegate didSaveSelector:(SEL)didSaveSelector contextInfo:(void
    *)contextInfo
```

Parameters

saveOperation
The type of save operation.

delegate
The delegate to which the selector message is sent.

didSaveSelector
The selector of the message sent to the delegate.

contextInfo
Object passed with the callback to provide any additional context information.

Discussion
When saving is completed, regardless of success or failure, or has been canceled, sends the message selected by *didSaveSelector* to the *delegate*, with *contextInfo* as the last argument. The method selected by *didSaveSelector* must have the same signature as:

```
- (void)document:(NSDocument *)doc didSave:(BOOL)didSave contextInfo:(void
*)contextInfo
```

Invoked from `saveDocumentWithDelegate:didSaveSelector:contextInfo:` (page 978), and the action methods `saveDocumentAs:` (page 977) and `saveDocumentTo:` (page 977). The default implementation of this method first makes sure that any editor registered using the Cocoa Bindings `NSEditorRegistration` informal protocol has committed its changes, then creates a Save panel, adds a standard file format accessory view (if there is more than one file type for the user to choose from and `shouldRunSavePanelWithAccessoryView` (page 987) returns `YES`), sets various attributes of the panel, invokes `prepareSavePanel:` (page 962) to provide an opportunity for customization, then presents the panel. If the user approves the panel, the default implementation sends the message `saveToURL:ofType:forSaveOperation:delegate:didSaveSelector:contextInfo:` (page 979).

For backward binary compatibility with Mac OS 10.3 and earlier, the default implementation of this method instead invokes the deprecated `saveToFile:saveOperation:delegate:didSaveSelector:contextInfo:` (page 978) if it is overridden, even if the user cancels the panel.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDocument.h`

runPageLayout:

The action method invoked in the receiver as first responder when the user chooses the Page Setup menu command.

```
- (void)runPageLayout:(id)sender
```

Parameters

sender

The control sending the message.

Discussion

The default implementation invokes `runModalPageLayoutWithPrintInfo:delegate:didRunSelector:contextInfo:` (page 973) with the document's current NSPrintInfo object as argument; if the user clicks the OK button and the document authorizes changes to its printing information (`shouldChangePrintInfo:` (page 985)), the method sets the document's new `NSPrintInfo` object and increments the document's change count.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setPrintInfo:` (page 984)
- `updateChangeCount:` (page 988)

Declared In

`NSDocument.h`

saveDocument:

The action method invoked in the receiver as first responder when the user chooses the Save menu command.

```
- (void)saveDocument:(id)sender
```

Parameters

sender

The control sending the message.

Discussion

The default implementation saves the document in two different ways, depending on whether the document has a file path and a document type assigned. If path and type are assigned, it simply writes the document under its current file path and type after making a backup copy of the previous file. If the document is new (no file path and type), it runs the modal Save panel to get the file location under which to save the document. It writes the document to this file, sets the document's file location and document type (if a native type), and clears the document's edited status.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `saveDocumentWithDelegate:didSaveSelector:contextInfo:` (page 978)
- `setFileName:` (page 981)
- `setFileType:` (page 982)
- `updateChangeCount:` (page 988)

Declared In

`NSDocument.h`

saveDocumentAs:

The action method invoked in the receiver as first responder when the user chooses the Save As menu command.

```
- (void)saveDocumentAs:(id)sender
```

Parameters

sender

The control sending the message.

Discussion

The default implementation runs the modal Save panel to get the file location under which to save the document. It writes the document to this file, sets the document's file location and document type (if a native type), and clears the document's edited status.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `saveDocumentWithDelegate:didSaveSelector:contextInfo:` (page 978)
- `setFileName:` (page 981)
- `setFileType:` (page 982)
- `updateChangeCount:` (page 988)

Declared In

`NSDocument.h`

saveDocumentTo:

The action method invoked in the receiver as first responder when the user chooses the Save To menu command.

```
- (void)saveDocumentTo:(id)sender
```

Parameters

sender

The control sending the message.

Discussion

The default implementation is identical to `saveDocumentAs:` except that this method doesn't clear the document's edited status and doesn't reset file location and document type if the document is a native type.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `saveDocumentWithDelegate:didSaveSelector:contextInfo:` (page 978)

Declared In

`NSDocument.h`

saveDocumentWithDelegate:didSaveSelector:contextInfo:

Saves the document.

```
- (void)saveDocumentWithDelegate:(id)delegate didSaveSelector:(SEL)didSaveSelector
    contextInfo:(void *)contextInfo
```

Parameters

delegate

The delegate to which the selector message is sent.

didSaveSelector

The selector of the message sent to the delegate.

contextInfo

Object passed with the callback to provide any additional context information.

Discussion

If an `NSSaveOperation` can be performed without further user intervention (at the very least, neither `fileURL` (page 951) nor `fileType` (page 950) return `nil`), then the method immediately saves the document. Otherwise, it presents a Save panel to the user and saves the document if the user approves the panel. When saving has been completed or canceled, the method sends the message selected by *didSaveSelector* to the *delegate*, with the *contextInfo* as the last argument.

As of Mac OS X v10.5, this method checks to see if the document's file has been modified since the document was opened or most recently saved or reverted, in addition to the checking for file moving, renaming, and trashing that it has done since Mac OS X v10.1. When it senses file modification it presents an alert telling the user "This document's file has been changed by another application since you opened or saved it," giving them the choice of saving or not saving. For backward binary compatibility this is only done in applications linked against Mac OS X v10.5 or later.

The *didSaveSelector* callback method should have the following signature:

```
- (void)document:(NSDocument *)doc didSave:(BOOL)didSave contextInfo:(void
*)contextInfo
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDocument.h`

saveToFile:saveOperation:delegate:didSaveSelector:contextInfo:

Called after the user has been given the opportunity to select a destination through the modal Save panel. (Deprecated in Mac OS X v10.4. Use `saveToURL:ofType:forSaveOperation:delegate:didSaveSelector:contextInfo:` (page 979) instead.)

```
- (void)saveToFile:(NSString *)fileName
    saveOperation:(NSSaveOperationType)saveOperation delegate:(id)delegate
    didSaveSelector:(SEL)didSaveSelector contextInfo:(void *)contextInfo
```

Discussion
Called after the user has been given the opportunity to select a destination through the modal Save panel presented by `runModalSavePanelForSaveOperation:delegate:didSaveSelector:contextInfo:` (page 975). The *`delegate`* is assigned to the Save panel. If *`fileName`* is non-`nil`, this method writes the document to *`fileName`*, sets the document's file location and document type (if a native type), and clears the document's edited status. *`didSaveSelector`* gets called with `YES` if the document is saved successfully, and `NO` otherwise. The *`saveOperation`* is one of the constants in "Constants" (page 999). Pass *`contextInfo`* with the callback.

The *`didSaveSelector`* callback method should have the following signature:

```
- (void)document:(NSDocument *)doc didSave:(BOOL)didSave contextInfo:(void
*)contextInfo
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`NSDocument.h`

saveToURL:ofType:forSaveOperation:delegate:didSaveSelector:contextInfo:

Saves the contents of the document to a file or file package located by a URL, formatted to a specified type, for a particular kind of save operation.

```
- (void)saveToURL:(NSURL *)absoluteURL ofType:(NSString *)typeName
    forSaveOperation:(NSSaveOperationType)saveOperation delegate:(id)delegate
    didSaveSelector:(SEL)didSaveSelector contextInfo:(void *)contextInfo
```

Parameters

`absoluteURL`
The location of the file or file package to which the document contents are saved.

`typeName`
The string that identifies the document type.

`saveOperation`
The type of save operation.

`delegate`
The delegate to which the selector message is sent.

`didSaveSelector`
The selector of the message sent to the delegate.

`contextInfo`
Object passed with the callback to provide any additional context information.

Discussion
When saving is completed, regardless of success or failure, the method sends the message selected by *`didSaveSelector`* to the *`delegate`*, with the *`contextInfo`* as the last argument. The method selected by *`didSaveSelector`* must have the same signature as:

```
- (void)document:(NSDocument *)document didSave:(BOOL)didSaveSuccessfully
contextInfo:(void  *)contextInfo;
```

The default implementation of this method invokes `[self saveToURL:absoluteURL ofType:typeName forSaveOperation:saveOperation error:&anError]` and, if `NO` is returned, presents the error to the user in a document-modal panel before messaging the delegate.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- saveToURL:ofType:forSaveOperation:error:` (page 980)

Declared In
`NSDocument.h`

saveToURL:ofType:forSaveOperation:error:

Saves the contents of the document to a file or file package located by a URL, formatted to a specified type, for a particular kind of save operation, and returns `YES` if successful.

```
- (BOOL)saveToURL:(NSURL *)absoluteURL ofType:(NSString *)typeName
    forSaveOperation:(NSSaveOperationType)saveOperation error:(NSError **)outError
```

Parameters

absoluteURL
The location of the file or file package to which the document contents are saved.

typeName
The string that identifies the document type.

saveOperation
The type of save operation.

outError
On return, If the document contents could not be saved, a pointer to an error object that encapsulates the reason they could not be saved.

Return Value
`YES` if the document contents were successfully saved; otherwise, `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- saveToURL:ofType:forSaveOperation:delegate:didSaveSelector:contextInfo:` (page 979)

Declared In
`NSDocument.h`

setAutosavedContentsFileURL:

Sets the location of the most recently autosaved document contents.

```
- (void)setAutosavedContentsFileURL:(NSURL *)absoluteURL
```

Parameters

absoluteURL

The location of the most recently autosaved document contents.

Discussion

The default implementation of this method records the URL and notifies the shared document controller that this document should be automatically reopened if the application quits or crashes before the document is saved.

Availability

Available in Mac OS X v10.4 and later.

See Also

- autosavedContentsFileURL (page 941)

Declared In

NSDocument.h

setFileModificationDate:

Sets the last known modification date of the document's on-disk representation to the given modification date.

```
- (void)setFileModificationDate:(NSDate *)modificationDate
```

Parameters

modificationDate

The date to which the file modification date is set.

Discussion

The NSDocument default file saving machinery uses this information to warn the user when the on-disk representation of an open document has been modified by something other than the current application.

Availability

Available in Mac OS X v10.4 and later.

See Also

- fileModificationDate (page 948)

Declared In

NSDocument.h

setFileName:

Sets the file (filename and directory path) under which document data is saved. (Deprecated in Mac OS X v10.4. Use setFileURL: (page 982) instead.)

```
- (void)setFileName:(NSString *)fileName
```

Discussion

Sets the file (filename and directory path) under which document data is saved to *fileName*. As a side effect, synchronizes the titles of the document's windows with the new name or location. A document's filename is automatically set when it is saved as a new document (Save) and when an existing document is saved

under a different filename or path (Save As). The Finder also keeps track of open documents and their associated files. When a user moves or renames a file in the Finder that corresponds to an open document, the Finder calls `setFileName:` with the new filename.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
`- fileName` (page 948)

Declared In
`NSDocument.h`

setFileType:

Sets the document type under which the file is saved.

```
- (void)setFileType:(NSString *)docType
```

Parameters
docType
The string that identifies the document type.

Discussion
The document type affects how the data is filtered when it is written to or read from a file. This method isn't for changing the document's format; it's just for initially recording the document's format during opening or saving.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- fileType` (page 950)

Related Sample Code
QTAudioExtractionPanel
QTKitImport
QTKitPlayer

Declared In
`NSDocument.h`

setFileURL:

Sets the location of the document's on-disk representation.

```
- (void)setFileURL:(NSURL *)absoluteURL
```

Parameters
absoluteURL
The document's location.

Discussion
This method doesn't actually rename the document; it's just for recording the document's location during initial opening or saving. The default implementation of this method just records the URL so that the default implementation of `fileURL` (page 951) can return it.

For backward binary compatibility with Mac OS X v10.3 and earlier, if `setFileName:` (page 981) is overridden and the URL is `nil` or uses the `file:` scheme, the default implementation of this method instead invokes `[self setFileName:[absoluteURL path]]`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `fileURL` (page 951)

Related Sample Code
FunHouse
QTMetadataEditor

Declared In
`NSDocument.h`

setHasUndoManager:

Sets whether the receiver has its own `NSUndoManager` object.

```
- (void)setHasUndoManager:(BOOL)flag
```

Parameters
flag
A Boolean value setting whether the receiver should own an `NSUndoManager` object.

Discussion
If *flag* is `NO` and the receiver currently owns an `NSUndoManager` object, the `NSUndoManager` object is released after being removed as an observer of undo-related notifications.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hasUndoManager` (page 954)

Declared In
`NSDocument.h`

setLastComponentOfFileName:

Sets the document name to the given string in terms of the scripting name property (the name a script writer would use to specify the document in a script).

```
- (void)setLastComponentOfFileName:(NSString *)str
```

Parameters

str

The scripting name of the document.

Discussion

Note that this name may be different than the name used in `setFileName:` (page 981).

Availability

Available in Mac OS X v10.0 and later.

See Also

`- displayName` (page 945)

Declared In

`NSDocumentScripting.h`

setPrintInfo:

Sets the receiver's `NSPrintInfo` object.

```
- (void)setPrintInfo:(NSPrintInfo *)printInfo
```

Parameters

printInfo

The `NSPrintInfo` object for the receiver to use.

Discussion

This `NSPrintInfo` object is used in laying out the document for printing.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- printInfo` (page 966)

Declared In

`NSDocument.h`

setUndoManager:

Sets the undo manager owned by the receiver to the specified undo manager and releases any undo manager currently owned by the receiver.

```
- (void)setUndoManager:(NSUndoManager *)undoManager
```

Parameters

undoManager

The undo manager to be owned by the receiver; may be `nil`.

Discussion

If *undoManager* is `nil`, it turns off the `hasUndoManager` flag. If *undoManager* is non-`nil`, it adds the receiver as an observer of `NSUndoManagerDidUndoChangeNotification`, `NSUndoManagerDidRedoChangeNotification`, and `NSUndoManagerWillCloseUndoGroupNotification`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- undoManager (page 988)

Declared In
NSDocument.h

setWindow:

Sets the window Interface Builder outlet of this class.

```
- (void)setWindow:(NSWindow *)aWindow
```

Parameters

aWindow

The window to which the receiver's window outlet points.

Discussion
This method is invoked automatically during the loading of any nib for which this document is the file's owner, if the file's owner window outlet is connected in the nib. You should not invoke this method directly, and typically you would not override it either.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDocument.h

shouldChangePrintInfo:

Returns a Boolean value indicating whether the receiver should allow changes to the default NSPrintInfo object used in printing the document.

```
- (BOOL)shouldChangePrintInfo:(NSPrintInfo *)newPrintInfo
```

Parameters

newPrintInfo

The NSPrintInfo object that is the result of the user approving the page layout panel presented by runPageLayout: (page 975).

Return Value
YES by default; subclasses can override this method to return NO.

Discussion
This method is invoked by the runPageLayout: (page 975) method, which sets a new NSPrintInfoobject for the document only if this method returns YES.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDocument.h

shouldCloseWindowController:

Gives the user an opportunity to save the document. (Available in Mac OS X v10.0 through Mac OS X v10.3. Use `shouldCloseWindowController:delegate:shouldCloseSelector:contextInfo:` (page 986) instead.)

```
- (BOOL)shouldCloseWindowController:(NSWindowController *)windowController
```

Discussion

If closing the *windowController* would cause the receiver to be closed, invokes `canCloseDocumentWithDelegate:shouldCloseSelector:contextInfo:` (page 943) to display a Save panel and give the user an opportunity to save the document. Returns the return value of `canCloseDocumentWithDelegate:shouldCloseSelector:contextInfo:`. Note that the receiver doesn't close until its window controller closes.

Availability

Available in Mac OS X v10.0 through Mac OS X v10.3.

Declared In

`NSDocument.h`

shouldCloseWindowController:delegate:shouldCloseSelector:contextInfo:

Invokes *shouldCloseSelector* with the result of `canCloseDocumentWithDelegate:shouldCloseSelector:contextInfo:` (page 943) if the the specified window controller that is closing is the last one or is marked as causing the document to close.

```
- (void)shouldCloseWindowController:(NSWindowController *)windowController
    delegate:(id)delegate shouldCloseSelector:(SEL)shouldCloseSelector
    contextInfo:(void *)contextInfo
```

Parameters

windowController
: The window controller that is closed.

delegate
: The delegate to which the selector message is sent.

shouldCloseSelector
: The selector of the message sent to the delegate.

contextInfo
: Object passed with the callback to provide any additional context information.

Discussion

Otherwise it invokes *shouldCloseSelector* with `YES`. This method is called automatically by `NSWindow` for any window that has a window controller and a document associated with it. `NSWindow` calls this method prior to sending its *delegate* the `windowShouldClose:` message. Pass the *contextInfo* object with the callback.

The *shouldCloseSelector* callback method should have the following signature:

```
- (void)document:(NSDocument *)document shouldClose:(BOOL)shouldClose
contextInfo:(void  *)contextInfo
```

Availability

Available in Mac OS X v10.0 and later.

Declared In
NSDocument.h

shouldRunSavePanelWithAccessoryView

Returns YES by default; as a result, when NSDocument displays the Save panel, it includes an accessory view containing a pop-up menu of supported writable document types.

```
- (BOOL)shouldRunSavePanelWithAccessoryView
```

Return Value
YES by default; subclasses can override to return NO, thereby excluding the accessory view from the Save panel.

Discussion
Here is an example implementation:

```
- (BOOL)shouldRunSavePanelWithAccessoryView {
    return [self fileName] == nil;
}
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- runModalSavePanelForSaveOperation:delegate:didSaveSelector:contextInfo: (page 975)

Declared In
NSDocument.h

showWindows

Displays all of the document's windows, bringing them to the front and making them main or key as necessary.

```
- (void)showWindows
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer
QTMetadataEditor

Declared In
NSDocument.h

undoManager

Returns the receiver's undo manager.

```
- (NSUndoManager *)undoManager
```

Return Value
The `NSUndoManager` object used by the receiver or `nil` if the receiver should not own one.

Discussion
If the undo manager doesn't exist and `hasUndoManager` returns `YES`, the method creates one and invokes `setUndoManager:` with the `NSUndoManager` as argument.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DemoMonkey
File Wrappers with Core Data Documents
FunHouse
Sketch+Accessibility
Sketch-112

Declared In
`NSDocument.h`

updateChangeCount:

Updates the receiver's change count according to the given change type.

```
- (void)updateChangeCount:(NSDocumentChangeType)changeType
```

Parameters
changeType
The type of change made to the document.

Discussion
The change count indicates the document's edited status; if the change count is 0, the document has no changes to save, and if the change count is greater than 0, the document has been edited and is unsaved. The *changeType* is described in "Constants" (page 999). If you are implementing undo and redo in an application, you should increment the change count every time you create an undo group and decrement the change count when an undo or redo operation is performed.

Note that if you are using the `NSDocument` default undo/redo features, setting the document's edited status by updating the change count happens automatically. You only need to invoke this method when you are not using these features.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitImport
QTKitPlayer

StillMotion

Declared In
NSDocument.h

validateMenuItem:

Validates the Revert menu item and items selected from the Save panel's pop-up list of writable document types items. (Available in Mac OS X v10.0 through Mac OS X v10.3. Use `validateUserInterfaceItem:` (page 989) instead.)

```
- (BOOL)validateMenuItem:(NSMenuItem *)anItem
```

Discussion
Returns `YES` if *anItem* should be enabled, `NO` otherwise. Returns `YES` for Revert if the document has been edited and a file exists for the document. Returns `YES` for an item representing a writable type if, during a Save or Save As operation, it is a native type for the document. Subclasses can override this method to perform additional validations.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.3.

Declared In
NSDocument.h

validateUserInterfaceItem:

Validates the specified user interface item that the receiver manages.

```
- (BOOL)validateUserInterfaceItem:(id < NSValidatedUserInterfaceItem >)anItem
```

Parameters
anItem
The user interface item to validate.

Return Value
`YES` if the item is valid; otherwise, `NO`.

Discussion
These items currently include only Revert (which is enabled only if the document has a `fileName` (page 948)) and Save. You can override this method to add more selectors validated by your document subclass.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDocument.h

willPresentError:

Called when the receiver is about to present an error.

```
- (NSError *)willPresentError:(NSError *)error
```

Parameters

error

The error object that is about to be presented to the user.

Return Value

The error that should actually be presented.

Discussion

The default implementation of this method merely returns the passed-in error. The returned error may simply be forwarded to the document controller.

You can override this method to customize the presentation of errors by examining the passed-in error and, for example, returning more specific information. When you override this method always check the `NSError` object's domain and code to discriminate between errors whose presentation you want to customize and those you don't. For errors you don't want to customize, call the superclass implementation, passing the original error.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `presentError:` (page 962)
- `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 963)

Declared In

`NSDocument.h`

windowControllerDidLoadNib:

Sent after the specified window controller loads a nib file if the receiver is the nib file's owner.

```
- (void)windowControllerDidLoadNib:(NSWindowController *)windowController
```

Parameters

windowController

The window controller that loads the nib file.

Discussion

See the class description for NSWindowController for additional information about nib files and the file's owner object.

Typically an `NSDocument` subclass overrides `windowNibName` (page 992) or `makeWindowControllers` (page 961), but not both. If `windowNibName` is overridden, the default implementation of `makeWindowControllers` will load the named nib file, making the `NSDocument` object the nib file's owner. In that case, you can override `windowControllerDidLoadNib:` and do custom processing after the nib file is loaded.

The default implementation of this method does nothing.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `windowControllerWillLoadNib:` (page 991)

- windowControllers (page 991)

Related Sample Code
ColorMatching
ThreadsExporter
ThreadsExportMovie
ThreadsImporter
ThreadsImportMovie

Declared In
NSDocument.h

windowControllers

Returns the receiver's current window controllers.

```
- (NSArray *)windowControllers
```

Return Value
An array containing NSWindowController objects belonging to the current document. If there are no window controllers, returns an empty NSArray object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- makeWindowControllers (page 961)
- windowControllerDidLoadNib: (page 990)
- windowControllerWillLoadNib: (page 991)
- windowNibName (page 992)

Related Sample Code
CoreTextArcCocoa
QTRecorder
Sketch-112

Declared In
NSDocument.h

windowControllerWillLoadNib:

Sent before the specified window controller loads a nib file if the receiver is the nib file's owner.

```
- (void)windowControllerWillLoadNib:(NSWindowController *)windowController
```

Parameters
windowController
The window controller that loads the nib file.

Discussion
See the class description for NSWindowController for additional information about nib files and the file's owner object.

Typically an `NSDocument` subclass overrides `windowNibName` (page 992) or `makeWindowControllers` (page 961), but not both. If `windowNibName` is overridden, the default implementation of `makeWindowControllers` will load the named nib file, making the NSDocument the nib file's owner. In that case, you can override `windowControllerWillLoadNib:` and do custom processing before the nib file is loaded.

The default implementation of this method does nothing.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `windowControllerDidLoadNib:` (page 990)
- `windowControllers` (page 991)

Declared In
`NSDocument.h`

windowForSheet

Returns the most appropriate window, of the windows associated with the receiver, to use as the parent window of a document-modal sheet.

```
- (NSWindow *)windowForSheet
```

Return Value
The window to use as the parent window of the sheet.

Discussion
May return `nil`, in which case the sender should present an application-modal panel. The `NSDocument` implementation of this method returns the window of the first window controller, or `[NSApp mainWindow]` if there are no window controllers or if the first window controller has no window.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
MovieAssembler
MyMediaPlayer

Declared In
`NSDocument.h`

windowNibName

Overridden by subclasses to return the name of the document's sole nib file.

```
- (NSString *)windowNibName
```

Return Value
The name of the document nib file.

Discussion
Using this name, `NSDocument` creates and instantiates a default instance of `NSWindowController` to manage the window. If your document has multiple nib files, each with its own single window, or if the default `NSWindowController` instance is not adequate for your purposes, you should override `makeWindowControllers`.

The default implementation returns `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- windowControllers` (page 991)

Related Sample Code
ImageApp
QTMetadataEditor
Simple Bindings Adoption
With and Without Bindings
ZipBrowser

Declared In
`NSDocument.h`

writableTypesForSaveOperation:

Returns the names of the types to which this document can be saved for a specified kind of save operation.

```
- (NSArray *)writableTypesForSaveOperation:(NSSaveOperationType)saveOperation
```

Parameters

saveOperation
The kind of save operation.

Return Value
An array of `NSString` objects representing the writable document types.

Discussion
The save operation type is represented by *saveOperation*. For every kind of save operation except `NSSaveToOperation`, the returned array must only include types for which the the application can play the Editor role. For `NSSaveToOperation` the returned array may include types for which the application can only play the Viewer role, and other types that the application can merely export. The default implementation of this method returns `[[self class] writableTypes]` with, except during `NSSaveToOperation`, types for which `isNativeType:` (page 939) returns `NO` filtered out.

You can override this method to limit the set of writable types when the document currently contains data that is not representable in all types. For example, you can disallow saving to RTF files when the document contains an attachment and can only be saved properly to RTFD files.

You can invoke this method when creating a custom save panel accessory view to present easily the same set of types as `NSDocument` does in its standard file format popup menu.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSDocument.h

writeSafelyToURL:ofType:forSaveOperation:error:

Writes the contents of the document to a file or file package located by a URL.

```
- (BOOL)writeSafelyToURL:(NSURL *)absoluteURL ofType:(NSString *)typeName
    forSaveOperation:(NSSaveOperationType)saveOperation error:(NSError **)outError
```

Parameters

absoluteURL
The location to which the document contents are written.

typeName
The string that identifies the document type.

saveOperation
The type of save operation.

outError
On return, If the document contents could not be written, a pointer to an error object that encapsulates the reason they could not be written.

Return Value
YES if the document contents could be written; otherwise, NO.

Discussion
The default implementation of this method invokes writeToURL:ofType:forSaveOperation:originalContentsURL:error: (page 997). It also invokes fileAttributesToWriteToURL:ofType:forSaveOperation:originalContentsURL:error: (page 946) and writes the returned attributes, if any, to the file. It may copy some attributes from the old on-disk revision of the document at the same time, if applicable.

This method is responsible for doing document writing in a way that minimizes the danger of leaving the disk to which writing is being done in an inconsistent state in the event of an application crash, system crash, hardware failure, power outage, and so on. If you override this method, be sure to invoke the superclass implementation.

For NSSaveOperation, the default implementation of this method invokes keepBackupFile (page 958) to determine whether or not the old on-disk revision of the document, if there was one, should be preserved after being renamed.

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method instead invokes writeWithBackupToFile:ofType:saveOperation: (page 998) if that method is is overridden and the URL uses the file: scheme. The save operation in this case is never NSAutosaveOperation; NSSaveToOperation is used instead.

Availability
Available in Mac OS X v10.4 and later.

See Also
- writeToURL:ofType:forSaveOperation:originalContentsURL:error: (page 997)

- fileAttributesToWriteToURL:ofType:forSaveOperation:originalContentsURL:error: (page 946)

Related Sample Code
QTMetadataEditor

Declared In
NSDocument.h

writeToFile:ofType:

Writes document data to a file. (Deprecated in Mac OS X v10.4. Use writeToURL:ofType:error: (page 996) instead.)

```
- (BOOL)writeToFile:(NSString *)fileName ofType:(NSString *)docType
```

Discussion
Writes document data of type *docType* to the file *fileName*, returning whether the operation was successful. This method invokes dataRepresentationOfType: (page 945) and is indirectly invoked whenever the document file is saved. It uses the NSData method writeToFile:atomically: to write to the file.

This method is one of the location-based primitives. Subclasses can override this method instead of overriding dataRepresentationOfType: to write document data to the file system as an NSData object after creating that object from internal data structures. Subclasses that handle file packages such as RTFD or that treat locations of files as anything other than paths should override this method. Override implementations of this method should ensure that they filter document data appropriately using NSPasteboard filtering services.

See "NSDocument Saving Behavior" (page 929) for additional information about saving documents.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- loadDataRepresentation:ofType: (page 959)
- readFromFile:ofType: (page 968)
- writeToFile:ofType:originalFile:saveOperation: (page 995)

Declared In
NSDocument.h

writeToFile:ofType:originalFile:saveOperation:

Writes the receiver document's contents to a file. (Deprecated in Mac OS X v10.4. Use writeToURL:ofType:forSaveOperation:originalContentsURL:error: (page 997) instead.)

```
- (BOOL)writeToFile:(NSString *)fullDocumentPath ofType:(NSString *)docType
    originalFile:(NSString *)fullOriginalDocumentPath
    saveOperation:(NSSaveOperationType)saveOperationType
```

Discussion
This method is called from `writeWithBackupToFile:ofType:saveOperation:` (page 998) to actually write the file of type *docType* to *fullDocumentPath*. *fullOriginalDocumentPath* is the path to the original file if there is one and `nil` otherwise. The default implementation simply calls `writeToFile:ofType:` (page 995). You should not need to call this method directly, but subclasses that need access to the previously saved copy of their document while saving the new one can override this method. The *saveOperationType* argument is one of the constants listed in "Constants" (page 999).

See "NSDocument Saving Behavior" (page 929) for additional information about saving documents.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`NSDocument.h`

writeToURL:ofType:

Writes document data to a URL. (Deprecated in Mac OS X v10.4. Use `writeToURL:ofType:error:` (page 996) instead.)

```
- (BOOL)writeToURL:(NSURL *)aURL ofType:(NSString *)docType
```

Discussion
Writes document data of type *docType* to the URL *aURL*, returning whether the operation was successful. This method only supports URLs of the `file:` scheme and calls `writeToFile:ofType:` (page 995).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`NSDocument.h`

writeToURL:ofType:error:

Writes the contents of the document to a file or file package located by a URL, formatted to a specified type.

```
- (BOOL)writeToURL:(NSURL *)absoluteURL ofType:(NSString *)typeName error:(NSError
    **)outError
```

Parameters

absoluteURL
The location to which the document contents are written.

typeName
The string that identifies the document type.

outError
On return, If the document contents could not be written, a pointer to an error object that encapsulates the reason they could not be written.

Return Value
`YES` if the document contents could be written; otherwise, `NO`.

Discussion
The default implementation of this method just invokes `[self fileWrapperOfType:typeName error:outError]` and writes the returned file wrapper to disk.

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method instead invokes `[self writeToFile:[absoluteURL path] ofType:typeName]` if `writeToFile:ofType:` (page 995) is overridden and the URL uses the `file:` scheme.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `fileWrapperOfType:error:` (page 951)
- `dataOfType:error:` (page 944)

Declared In
`NSDocument.h`

writeToURL:ofType:forSaveOperation:originalContentsURL:error:

Writes the contents of the document to a file or file package located by a URL.

```
- (BOOL)writeToURL:(NSURL *)absoluteURL ofType:(NSString *)typeName
    forSaveOperation:(NSSaveOperationType)saveOperation originalContentsURL:(NSURL
     *)absoluteOriginalContentsURL error:(NSError **)outError
```

Parameters

absoluteURL
The location to which the document contents are written.

typeName
The string that identifies the document type.

saveOperation
The type of save operation.

absoluteOriginalContentsURL
The location of the previously saved copy of the document (if not `nil`).

outError
On return, If the document contents could not be written, a pointer to an error object that encapsulates the reason they could not be written.

Return Value
`YES` if the document contents could be written; otherwise, `NO`.

Discussion
The default implementation of this method merely invokes `[self writeToURL:absoluteURL ofType:typeName error:outError]`. You can override this method instead of one of the three simple writing methods (`writeToURL:ofType:error:` (page 996), `fileWrapperOfType:error:` (page 951), and `dataOfType:error:` (page 944)) if your document writing machinery needs access to the on-disk representation of the document revision that is about to be overwritten. The value of *absoluteURL* is often not the same as `[self fileURL]`. Other times it is not the same as the URL for the final save destination.

Likewise, *absoluteOriginalContentsURL* is often not the same value as `[self fileURL]`. If *absoluteOriginalContentsURL* is `nil`, either the document has never been saved or the user deleted the document file since it was opened.

For backward binary compatibility with Mac OS X v10.3 and earlier, if `writeToFile:ofType:originalFile:saveOperation:` (page 995) is overridden and both URLs use the `file:` scheme, the default implementation of this method instead invokes:

```
[self writeToFile:[absoluteURL path]
      ofType:typeName
      originalFile:[absoluteOriginalContentsURL path]
      saveOperation:aSaveOperation];
```

The save operation used in this case is never `NSAutosaveOperation`; `NSSaveToOperation` is used instead.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSDocument.h`

writeWithBackupToFile:ofType:saveOperation:

This method is called by action methods to save document contents to a file. (Deprecated in Mac OS X v10.4. Use `writeSafelyToURL:ofType:forSaveOperation:error:` (page 994) instead.)

```
- (BOOL)writeWithBackupToFile:(NSString *)fullDocumentPath ofType:(NSString *)docType
    saveOperation:(NSSaveOperationType)saveOperationType
```

Discussion
This method is called by action methods like `saveDocument:` (page 976), `saveDocumentAs:` (page 977), and `saveDocumentTo:` (page 977). It is responsible for handling backup of the existing file, if any, and removal of that backup if `keepBackupFile` (page 958) returns `NO`. In between those two things, it calls `writeToFile:ofType:originalFile:saveOperation:` (page 995) to write the document of type *docType* to *fullDocumentPath*. You should never need to call `writeWithBackupToFile:ofType:saveOperation:`, but subclasses that want to change the way the backup works can override it. The *saveOperationType* argument is one of the constants listed in "Constants" (page 999).

If you override this method, you should invoke `fileAttributesToWriteToFile:ofType:saveOperation:` (page 946) and set the variables returned from this method when writing *fullDocumentPath*. `NSFileManager changeFileAttributes:atPath:` can be used to do this.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer

Declared In
NSDocument.h

Constants

NSSaveOperationType

The following constants specify types of save operations. These values are used with method parameters of type NSSaveOperationType. Depending on the method, those parameters can affect the title of the Save panel, as well as the files displayed.

```
enum {
    NSSaveOperation   = 0,
    NSSaveAsOperation = 1,
    NSSaveToOperation = 2
    NSAutosaveOperation = 3
};
typedef NSUInteger NSSaveOperationType;
```

Constants

NSSaveOperation
: Specifies a Save operation, the overwriting of a document's file or file package with the document's current contents.

  Available in Mac OS X v10.0 and later.

  Declared in NSDocument.h.

NSSaveAsOperation
: Specifies a Save As operation, the writing of a document's current contents to a new file or file package, and then making the just-written file or file package the document's current one.

  Available in Mac OS X v10.0 and later.

  Declared in NSDocument.h.

NSSaveToOperation
: Specifies a Save To operation, the writing of a document's current contents to a new file or file package without changing the document's current one.

  Available in Mac OS X v10.0 and later.

  Declared in NSDocument.h.

NSAutosaveOperation
: Specifies an autosave operation, writing a document’s contents to a file or file package separate from the document's current one, without changing the document's current one.

  Available in Mac OS X v10.4 and later.

  Declared in NSDocument.h.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDocument.h

NSDocumentChangeType

Change counts indicate a document's edit status. These constants indicate how a document should operate on its change count and are passed to `updateChangeCount:` (page 988).

```
enum {
    NSChangeDone = 0,
    NSChangeUndone = 1,
    NSChangeCleared = 2,
    NSChangeReadOtherContents = 3,
    NSChangeAutosaved = 4,
    NSChangeRedone = 5
};
typedef NSUInteger NSDocumentChangeType;
```

Constants

`NSChangeDone`

Increment change count. The value to pass to `updateChangeCount:` (page 988) to indicate that a single change has been done. For example, the built-in undo support of `NSDocument` passes this value whenever a document receives an `NSUndoManagerWillCloseUndoGroupNotification` from its own undo manager.

Available in Mac OS X v10.0 and later.

Declared in `NSDocument.h`.

`NSChangeUndone`

Decrement change count. A single change has been undone. For example, the built-in undo support of `NSDocument` passes this value whenever a document receives an `NSUndoManagerDidUndoChangeNotification` from its own undo manager.

Available in Mac OS X v10.0 and later.

Declared in `NSDocument.h`.

`NSChangeCleared`

Set change count to 0. The document has been synchronized with its file or file package. For example, `saveToURL:ofType:forSaveOperation:error:` (page 980) passes this value for a successful `NSSaveOperation` (page 999) or `NSSaveAsOperation` (page 999). The `revertDocumentToSaved:` (page 971) method does too.

Available in Mac OS X v10.0 and later.

Declared in `NSDocument.h`.

`NSChangeReadOtherContents`

The document has been initialized with the contents of a file or file package other than the one whose location would be returned by `fileURL` (page 951), and therefore can't possibly be synchronized with its persistent representation. For example, `initForURL:withContentsOfURL:ofType:error:` (page 955) passes this value when the two passed-in URLs are not equal to indicate that an autosaved document is being reopened.

Available in Mac OS X v10.4 and later.

Declared in `NSDocument.h`.

`NSChangeAutosaved`

The document's contents have been autosaved. For example, `saveToURL:ofType:forSaveOperation:error:` (page 980) passes this value for a successful `NSAutosaveOperation` (page 999).

Available in Mac OS X v10.4 and later.

Declared in `NSDocument.h`.

`NSChangeRedone`

A single change has been redone. For example, the built-in undo support of `NSDocument` passes this value whenever a document receives an `NSUndoManagerDidRedoChangeNotification` from its own undo manager.

Available in Mac OS X v10.5 and later.

Declared in `NSDocument.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDocument.h`

NSDocumentController Class Reference

Inherits from	NSObject
Conforms to	NSUserInterfaceValidations NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSDocumentController.h
Companion guide	Document-Based Applications Overview
Related sample code	CIRAWFilterSample QTAudioContextInsert QTAudioExtractionPanel QTKitPlayer Quartz Composer WWDC 2005 TextEdit

Overview

An `NSDocumentController` object manages an application's documents. As the first-responder target of New and Open menu commands, it creates and opens documents and tracks them throughout a session of the application. When opening documents, an `NSDocumentController` runs and manages the modal Open panel. `NSDocumentController` objects also maintain and manage the mappings of document types, extensions, and `NSDocument` subclasses as specified in the `CFBundleDocumentTypes` property loaded from the information property list (`Info.plist`).

You can use various `NSDocumentController` methods to get a list of the current documents; get the current document (which is the document whose window is currently key); get documents based on a given filename or window; and find out about a document's extension, type, display name, and document class.

In some situations, it is worthwhile to subclass `NSDocumentController` in non-`NSDocument`-based applications to get some of its features. For example, the `NSDocumentController` management of the Open Recent menu is useful in applications that don't use subclasses of `NSDocument`.

Adopted Protocols

NSCoding

`encodeWithCoder:`
`initWithCoder:`

Tasks

Obtaining the Shared Document Controller

`+ sharedDocumentController` (page 1008)
Returns the shared `NSDocumentController` instance.

Initializing a New NSDocumentController

`- init` (page 1017)
This method is the designated initializer for `NSDocumentController`.

Creating and Opening Documents

`- documentForURL:` (page 1014)
Returns, for a given URL, the open document whose file or file package is located by the URL, or `nil` if there is no such open document.

`- openUntitledDocumentAndDisplay:error:` (page 1024)
Creates a new untitled document, presents its user interface if *`displayDocument`* is `YES`, and returns the document if successful.

`- makeUntitledDocumentOfType:error:` (page 1020)
Instantiates a new untitled document of the specified type and returns it if successful.

`- openDocumentWithContentsOfURL:display:error:` (page 1023)
Opens a document located by the given URL presents its user interface if requested, and returns the document if successful.

`- makeDocumentWithContentsOfURL:ofType:error:` (page 1019)
Instantiates a document located by a URL, of a specified type, and returns it if successful.

`- reopenDocumentForURL:withContentsOfURL:error:` (page 1027)
Reopens an autosaved document located by a URL, by reading the contents for the document from another URL, presents its user interface, and returns `YES` if successful.

`- makeDocumentForURL:withContentsOfURL:ofType:error:` (page 1017)
Instantiates a document located by a URL, of a specified type, but by reading the contents for the document from another URL, and returns it if successful.

Managing Documents

`- documents` (page 1015)
Returns the `NSDocument` objects managed by the receiver.

- `addDocument:` (page 1009)
  Adds the given document to the list of open documents.
- `currentDocument` (page 1011)
  Returns the `NSDocument` object associated with the main window.
- `documentForWindow:` (page 1014)
  Returns the document object whose window controller owns a specified window.
- `hasEditedDocuments` (page 1016)
  Returns a Boolean value that indicates whether the receiver has any documents with unsaved changes.
- `removeDocument:` (page 1026)
  Removes the given document from the list of open documents.

Managing Document Types

- `documentClassNames` (page 1013)
  Returns the names of `NSDocument` subclasses supported by this application.
- `defaultType` (page 1011)
  Returns the name of the document type that should be used when creating new documents.
- `documentClassForType:` (page 1012)
  Returns the `NSDocument` subclass associated with a given document type.
- `displayNameForType:` (page 1012)
  Returns the descriptive name for the specified document type, which is used in the File Format pop-up menu of the Save As dialog.
- `typeForContentsOfURL:error:` (page 1030)
  Returns, for a specified URL, the name of the document type that should be used when opening the document at that location, if successful.
- `fileExtensionsFromType:` (page 1015) Deprecated in Mac OS X v10.5
  Returns the allowable file extensions for the given document type. (Deprecated. Use the `NSDocument` method `fileNameExtensionForType:saveOperation:` (page 949) instead.)
- `typeFromFileExtension:` (page 1030) Deprecated in Mac OS X v10.5
  Returns the document type associated with files having extension *`fileExtensionOrHFSFileType`*. (Deprecated. Use `typeForContentsOfURL:error:` (page 1030) instead.)

Autosaving

- `autosavingDelay` (page 1009)
  Returns the time interval in seconds for periodic autosaving.
- `setAutosavingDelay:` (page 1028)
  Sets the time interval in seconds for periodic autosaving.

Closing Documents

- `closeAllDocumentsWithDelegate:didCloseAllSelector:contextInfo:` (page 1010)
  Iterates through all the open documents and tries to close them one by one using the specified delegate.

- reviewUnsavedDocumentsWithAlertTitle:cancellable:delegate:didReviewAllSelector:contextInfo: (page 1027)

  Displays an alert dialog asking if the user wants to review unsaved documents (only if there are two or more unsaved documents), quit regardless of unsaved documents, or (if the choice is allowed) cancel the impending save operation.

Responding to Action Messages

- newDocument: (page 1020)

  An action method invoked by the New menu command, this method creates a new NSDocument object and adds it to the list of such objects managed by the document controller.

- openDocument: (page 1021)

  An action method invoked by the Open menu command, it runs the modal Open panel and, based on the selected filenames, creates one or more NSDocument objects from the contents of the files.

- saveAllDocuments: (page 1028)

  As the action method invoked by the Save All command, saves all open documents of the application that need to be saved.

Managing the Open Panel

- runModalOpenPanel:forTypes: (page 1028)

  Invokes the NSOpenPanel runModalForTypes: (page 1821) method, passing the *openPanel* object and the file *extensions* associated with a document type.

- currentDirectory (page 1010)

  Returns the directory path to be used as the starting point in the Open panel.

- URLsFromRunningOpenPanel (page 1031)

  Creates an NSOpenPanel instance and initializes it appropriately.

Managing the Open Recent Menu

- maximumRecentDocumentCount (page 1020)

  Returns the maximum number of items that may be presented in the standard Open Recent menu.

- clearRecentDocuments: (page 1009)

  Empties the recent documents list for the application.

- noteNewRecentDocumentURL: (page 1021)

  This method should be called by applications not based on NSDocument when they open or save documents identified by the given URL.

- noteNewRecentDocument: (page 1021)

  This method is called by NSDocument objects at appropriate times for managing the recent-documents list.

- recentDocumentURLs (page 1026)

  Returns the list of recent-document URLs.

Validating User Interface Items

- `validateUserInterfaceItem:` (page 1031)
  Returns a Boolean value that indicates whether a given user interface item should be enabled.

Handling Errors

- `presentError:` (page 1025)
  Presents an error alert to the user as a modal panel.
- `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 1026)
  Presents an error alert to the user as a modal panel.
- `willPresentError:` (page 1031)
  Called when the receiver is about to present an error, returns the error that should actually be presented.

Deprecated Methods

- `documentForFileName:` (page 1013) Deprecated in Mac OS X v10.4
  Returns the document object for the file in which the document data is stored. (Deprecated. Use `documentForURL:` (page 1014) instead.)
- `fileNamesFromRunningOpenPanel` (page 1016) Deprecated in Mac OS X v10.4
  Returns a selection of files chosen by the user in the Open panel. (Deprecated. Use `URLsFromRunningOpenPanel` (page 1031) instead.)
- `makeDocumentWithContentsOfFile:ofType:` (page 1017) Deprecated in Mac OS X v10.4
  Creates and returns a document object of a given document type from the contents of a file. (Deprecated. Use `makeDocumentWithContentsOfURL:ofType:error:` (page 1019) instead.)
- `makeDocumentWithContentsOfURL:ofType:` (page 1018) Deprecated in Mac OS X v10.4
  Creates and returns a ocument object for the given document type from the contents of a given URL. (Deprecated. Use `makeDocumentWithContentsOfURL:ofType:error:` (page 1019) instead.)
- `makeUntitledDocumentOfType:` (page 1019) Deprecated in Mac OS X v10.4
  Creates and returns a document object for document type. (Deprecated. Use `makeUntitledDocumentOfType:error:` (page 1020) instead.)
- `openDocumentWithContentsOfFile:display:` (page 1022) Deprecated in Mac OS X v10.4
  Returns a document object created from the contents of a given file and optionally displays it. (Deprecated. Use `openDocumentWithContentsOfURL:display:error:` (page 1023) instead.)
- `openDocumentWithContentsOfURL:display:` (page 1023) Deprecated in Mac OS X v10.4
  Returns a document object created from the contents of a given URL and optionally displays it. (Deprecated. Use `openDocumentWithContentsOfURL:display:error:` (page 1023) instead.)
- `openUntitledDocumentOfType:display:` (page 1024) Deprecated in Mac OS X v10.4
  Returns a document object instantiated from the subclass of the given document type and optionally displays it. (Deprecated. Use `openUntitledDocumentAndDisplay:error:` (page 1024) with `defaultType` (page 1011) instead.)

- `setShouldCreateUI:` (page 1029) Deprecated in Mac OS X v10.4
  Sets whether the window controllers of a document should be created when the document is created. (Deprecated. Use the display parameter of `openUntitledDocumentAndDisplay:error:` (page 1024) or `openDocumentWithContentsOfURL:display:error:` (page 1023) instead.)
- `shouldCreateUI` (page 1029) Deprecated in Mac OS X v10.4
  Returns a Boolean value that indicates whether the window controllers of a document should be created when the document is created. (Deprecated. Use the *display* parameter of `openUntitledDocumentAndDisplay:error:` (page 1024) or `openDocumentWithContentsOfURL:display:error:` (page 1023) instead.)

Class Methods

sharedDocumentController

Returns the shared `NSDocumentController` instance.

```
+ (id)sharedDocumentController
```

Return Value
The shared `NSDocumentController` instance.

Discussion
If an `NSDocumentController` instance doesn't exist yet, it is created.

Initialization reads in the document types from the `CFBundleDocumentTypes` property list (in `Info.plist`), registers the instance for `NSWorkspaceWillPowerOffNotification` (page 3495)s, and turns on the flag indicating that document user interfaces should be visible. You should always obtain your application's `NSDocumentController` using this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setShouldCreateUI:` (page 1029)

Related Sample Code
FunHouse
QTAudioContextInsert
QTAudioExtractionPanel
QTKitPlayer
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSDocumentController.h`

Instance Methods

addDocument:

Adds the given document to the list of open documents.

```
- (void)addDocument:(NSDocument *)document
```

Discussion
The `open...` methods automatically call `addDocument:`. This method is mostly provided for subclasses that want to know when documents arrive.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer
QTMetadataEditor

Declared In
`NSDocumentController.h`

autosavingDelay

Returns the time interval in seconds for periodic autosaving.

```
- (NSTimeInterval)autosavingDelay
```

Discussion
A value of 0 indicates that periodic autosaving should not be done at all. `NSDocumentController` uses this number as the amount of time to wait between detecting that a document has unautosaved changes and sending the document an `autosaveDocumentWithDelegate:didAutosaveSelector:contextInfo:` (page 941) message. The default value is 0.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAutosavingDelay:` (page 1028)

Declared In
`NSDocumentController.h`

clearRecentDocuments:

Empties the recent documents list for the application.

```
- (IBAction)clearRecentDocuments:(id)sender
```

Discussion
This is the action for the Clear menu command, but it can be invoked directly if necessary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDocumentController.h`

closeAllDocumentsWithDelegate:didCloseAllSelector:contextInfo:

Iterates through all the open documents and tries to close them one by one using the specified delegate.

```
- (void)closeAllDocumentsWithDelegate:(id)delegate
    didCloseAllSelector:(SEL)didCloseAllSelector contextInfo:(void *)contextInfo
```

Discussion
Each `NSDocument` object is sent `canCloseDocumentWithDelegate:shouldCloseSelector:contextInfo:` (page 943), which, if the document is dirty, gives it a chance to refuse to close or to save itself first. This method may ask whether to save or to perform a save.

The *`didCloseAllSelector`* callback method is invoked with `YES` if all documents are closed, and `NO` otherwise. Pass the *`contextInfo`* object with the callback. The *`didCloseAllSelector`* callback method should have the following signature:

```
- (void)documentController:(NSDocumentController *)docController  didCloseAll:
 (BOOL)didCloseAll contextInfo:(void *)contextInfo
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDocumentController.h`

currentDirectory

Returns the directory path to be used as the starting point in the Open panel.

```
- (NSString *)currentDirectory
```

Discussion
The first valid directory from the following list is returned:

- The directory location where the current document was last saved
- The last directory visited in the Open panel
- The user's home directory

Availability
Available in Mac OS X v10.0 and later.

See Also
- `documentForFileName:` (page 1013)

Declared In
`NSDocumentController.h`

currentDocument

Returns the `NSDocument` object associated with the main window.

```
- (id)currentDocument
```

Discussion
This method returns `nil` if it is called when its application is not active. This can occur during processing of a drag-and-drop operation, for example, in an implementation of `readSelectionFromPasteboard:`. In such a case, send the following message instead from an `NSView` subclass associated with the document:

```
[[[self window] windowController] document];
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `documentForFileName:` (page 1013)
- `documentForWindow:` (page 1014)
- `documents` (page 1015)

Related Sample Code
FinalCutPro_AppleEvents
FunHouse
QTAudioContextInsert
QTAudioExtractionPanel
QTKitPlayer

Declared In
`NSDocumentController.h`

defaultType

Returns the name of the document type that should be used when creating new documents.

```
- (NSString *)defaultType
```

Discussion
The default implementation of this method returns the first Editor type declared by the `CFBundleDocumentTypes` array in the application's `Info.plist`, or returns `nil` if no Editor type is declared. You can override it to customize the type of document that is created when, for instance, the user chooses New in the File menu.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
CIRAWFilterSample
ImageApp

Declared In
NSDocumentController.h

displayNameForType:

Returns the descriptive name for the specified document type, which is used in the File Format pop-up menu of the Save As dialog.

```
- (NSString *)displayNameForType:(NSString *)documentTypeName
```

Parameters

documentTypeName

The name of a document type, specified by `CFBundleTypeName` in the application's `Info.plist` file.

Return Value
The descriptive name for the document type specified by *documentTypeName*. If there is no descriptive name, returns *documentTypeName*.

Discussion
For a document-based application, supported document types are specified in the `Info.plist` file by the `CFBundleDocumentTypes` array. Each document type is specified by a dictionary in this array, and is named by the `CFBundleTypeName` attribute. You can provide a descriptive, localized, representation of this name by providing a corresponding entry in the `InfoPlist.strings` file(s). For example, given an `Info.plist` file that contains the following fragment:

```
<dict>
    <key>CFBundleDocumentTypes</key>
    <array>
        <dict>
            <key>CFBundleTypeName</key>
            <string>BinaryFile</string>
            <key>CFBundleTypeExtensions</key>
            <array>
                <string>binary</string>
            </array>
```

you could provide a descriptive name by adding an entry in the `InfoPlist.strings` file:

```
BinaryFile = "Binary file format";
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDocumentController.h

documentClassForType:

Returns the `NSDocument` subclass associated with a given document type.

```
- (Class)documentClassForType:(NSString *)documentTypeName
```

Parameters

documentTypeName

The name of a document type, specified by `CFBundleTypeName` in the application's `Info.plist` file.

The document type must be one the receiver can read.

Return Value
Returns the `NSDocument` subclass associated with *documentTypeName*. If the class cannot be found, returns `nil`.

Discussion
Para

Availability
Available in Mac OS X v10.0 and later.

See Also
`- displayNameForType:` (page 1012)

Declared In
`NSDocumentController.h`

documentClassNames

Returns the names of `NSDocument` subclasses supported by this application.

```
- (NSArray *)documentClassNames
```

Return Value
The names of `NSDocument` subclasses supported by this application.

Discussion
The default implementation of this method returns information derived from the application's `Info.plist` property list file. You can override it to return the names of document classes that are dynamically loaded from plugins.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
ImageApp

Declared In
`NSDocumentController.h`

documentForFileName:

Returns the document object for the file in which the document data is stored. (Deprecated in Mac OS X v10.4. Use `documentForURL:` (page 1014) instead.)

```
- (id)documentForFileName:(NSString *)fileName
```

Discussion
The *fileName* argument is a fully qualified path in the file system. Returns `nil` if no document can be found.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- `documentForWindow:` (page 1014)
- `documents` (page 1015)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitAdvancedDocument
QTKitImport
QTKitPlayer

Declared In
`NSDocumentController.h`

documentForURL:

Returns, for a given URL, the open document whose file or file package is located by the URL, or `nil` if there is no such open document.

```
- (id)documentForURL:(NSURL *)absoluteURL
```

Discussion
The default implementation of this method queries each open document to find one whose URL matches, and returns the first one whose URL does match.

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method instead invokes `documentForFileName:` (page 1013) if it is overridden and the URL uses the `file:` scheme.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
QTKitCreateMovie
QTKitFrameStepper
QTMetadataEditor

Declared In
`NSDocumentController.h`

documentForWindow:

Returns the document object whose window controller owns a specified window.

```
- (id)documentForWindow:(NSWindow *)window
```

Return Value
The document object whose window controller owns *window*. Returns `nil` if *window* is `nil`, if *window* has no window controller, or if the window controller does not have an association with an instance of `NSDocument`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `currentDocument` (page 1011)
- `documentForFileName:` (page 1013)
- `documents` (page 1015)

Related Sample Code
ImageApp

Declared In
`NSDocumentController.h`

documents

Returns the `NSDocument` objects managed by the receiver.

```
- (NSArray *)documents
```

Return Value
The `NSDocument` objects managed by the receiver. If there are currently no documents, returns an empty `NSArray` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `currentDocument` (page 1011)
- `documentForFileName:` (page 1013)
- `documentForWindow:` (page 1014)

Related Sample Code
DemoMonkey
QTAudioExtractionPanel

Declared In
`NSDocumentController.h`

fileExtensionsFromType:

Returns the allowable file extensions for the given document type. (Deprecated in Mac OS X v10.5. Use the `NSDocument` method `fileNameExtensionForType:saveOperation:` (page 949) instead.)

```
- (NSArray *)fileExtensionsFromType:(NSString *)documentTypeName
```

Parameters

documentTypeName

The name of a document type, specified by `CFBundleTypeName` in the application's `Info.plist` file.

Return Value

The allowable file extensions (as `NSString` objects) for *documentTypeName*.

Discussion

Type extensions are specified by the `CFBundleTypeExtensions` array for the given type in the `Info.plist` file.

The first string in the returned array is typically the most common extension. The array may also contain encoded HFS file types as will as filename extensions.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

See Also

`- displayNameForType:` (page 1012)

Declared In

`NSDocumentController.h`

fileNamesFromRunningOpenPanel

Returns a selection of files chosen by the user in the Open panel. (Deprecated in Mac OS X v10.4. Use `URLsFromRunningOpenPanel` (page 1031) instead.)

```
- (NSArray *)fileNamesFromRunningOpenPanel
```

Discussion

Each file in the returned `NSArray` is a fully qualified path to the file's location in the file system. This method is invoked by `openDocument:` (page 1021), and it invokes `runModalOpenPanel:forTypes:` (page 1028) after initializing the Open panel (which includes getting the starting directory with `currentDirectory` (page 1010)). Returns `nil` if the user cancels the Open panel or makes no selection.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`NSDocumentController.h`

hasEditedDocuments

Returns a Boolean value that indicates whether the receiver has any documents with unsaved changes.

```
- (BOOL)hasEditedDocuments
```

Return Value

`YES` if the receiver has any documents with unsaved changes, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `documents` (page 1015)

Declared In
`NSDocumentController.h`

init

This method is the designated initializer for `NSDocumentController`.

```
- (id)init
```

Discussion
The first instance of `NSDocumentController` or any of its subclasses that is created becomes the shared instance.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDocumentController.h`

makeDocumentForURL:withContentsOfURL:ofType:error:

Instantiates a document located by a URL, of a specified type, but by reading the contents for the document from another URL, and returns it if successful.

```
- (id)makeDocumentForURL:(NSURL *)absoluteDocumentURL withContentsOfURL:(NSURL
    *)absoluteDocumentContentsURL ofType:(NSString *)typeName error:(NSError
    **)outError
```

Discussion
The URL is specified by *absoluteDocumentURL*, the type by *typeName*, and the other URL providing the contents by *absoluteDocumentContentsURL*. If not successful, the method returns `nil` after setting *outError* to point to an `NSError` object that encapsulates the reason why the document could not be instantiated. The default implementation of this method invokes `documentClassForType:` (page 1012) to find out the class of document to instantiate, allocates a document object, and initializes it by sending it an `initForURL:withContentsOfURL:ofType:error:` (page 955) message.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSDocumentController.h`

makeDocumentWithContentsOfFile:ofType:

Creates and returns a document object of a given document type from the contents of a file. (Deprecated in Mac OS X v10.4. Use `makeDocumentWithContentsOfURL:ofType:error:` (page 1019) instead.)

```
- (id)makeDocumentWithContentsOfFile:(NSString *)fileName ofType:(NSString *)docType
```

Discussion
Creates and returns an `NSDocument` object for document type *docType* from the contents of the file *fileName*, which must be a fully qualified path. The returned object is not retained. Returns `nil` if the `NSDocument` subclass for *docType* couldn't be determined or if the object couldn't be created. This method invokes the `NSDocument` method `initWithContentsOfFile:ofType:` (page 956) and is invoked by `openDocumentWithContentsOfFile:display:` (page 1022).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- `makeUntitledDocumentOfType:` (page 1019)
- `openDocument:` (page 1021)

Related Sample Code
QTKitImport

Declared In
`NSDocumentController.h`

makeDocumentWithContentsOfURL:ofType:

Creates and returns a ocument object for the given document type from the contents of a given URL. (Deprecated in Mac OS X v10.4. Use `makeDocumentWithContentsOfURL:ofType:error:` (page 1019) instead.)

```
- (id)makeDocumentWithContentsOfURL:(NSURL *)aURL ofType:(NSString *)docType
```

Discussion
Creates and returns an `NSDocument` object for document type *docType* from the contents of *aURL*. The returned object is not retained. Returns `nil` if the `NSDocument` subclass for *docType* couldn't be determined or if the object couldn't be created. This method invokes the `NSDocument` method `initWithContentsOfURL:ofType:` (page 956) and is invoked by `openDocumentWithContentsOfURL:display:` (page 1023).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- `makeUntitledDocumentOfType:` (page 1019)
- `makeUntitledDocumentOfType:error:` (page 1020)
- `openDocument:` (page 1021)

Related Sample Code
QTAudioExtractionPanel

Declared In
`NSDocumentController.h`

makeDocumentWithContentsOfURL:ofType:error:

Instantiates a document located by a URL, of a specified type, and returns it if successful.

```
- (id)makeDocumentWithContentsOfURL:(NSURL *)absoluteURL ofType:(NSString *)typeName
    error:(NSError **)outError
```

Discussion
The URL is specified by *absoluteURL* and the document type by *typeName*. If not successful, the method returns `nil` after setting *outError* to point to an NSError that encapsulates the reason why the document could not be instantiated. The default implementation of this method invokes `documentClassForType:` (page 1012) to find out the class of document to instantiate, allocates a document object, and initializes it by sending it an `initWithContentsOfURL:ofType:error:` (page 957) message.

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method instead invokes `makeDocumentWithContentsOfFile:ofType:` (page 1017) if it is overridden and the URL uses the `file:` scheme.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
QTKitPlayer

Declared In
`NSDocumentController.h`

makeUntitledDocumentOfType:

Creates and returns a document object for document type. (Deprecated in Mac OS X v10.4. Use `makeUntitledDocumentOfType:error:` (page 1020) instead.)

```
- (id)makeUntitledDocumentOfType:(NSString *)type
```

Discussion
Creates and returns an `NSDocument` object for document type *type*. The returned object is not retained. Returns `nil` if the `NSDocument` subclass for *type* couldn't be determined or if the object couldn't be created. This method invokes the `NSDocument` `init` (page 954) method and is invoked by `openUntitledDocumentOfType:display:` (page 1024).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- `makeDocumentWithContentsOfFile:ofType:` (page 1017)
- `newDocument:` (page 1020)

Declared In
`NSDocumentController.h`

makeUntitledDocumentOfType:error:

Instantiates a new untitled document of the specified type and returns it if successful.

```
- (id)makeUntitledDocumentOfType:(NSString *)typeName error:(NSError **)outError
```

Discussion
The document type is specified by *typeName*. If not successful, the method returns `nil` after setting *outError* to point to an `NSError` object that encapsulates the reason why a new untitled document could not be instantiated. The default implementation of this method invokes `documentClassForType:` (page 1012) to find out the class of document to instantiate, then allocates and initializes a document by sending it `initWithType:error:` (page 957).

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method instead invokes `makeUntitledDocumentOfType:` (page 1019) if it is overridden.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSDocumentController.h`

maximumRecentDocumentCount

Returns the maximum number of items that may be presented in the standard Open Recent menu.

```
- (NSUInteger)maximumRecentDocumentCount
```

Discussion
A value of 0 indicates that `NSDocumentController` will not attempt to add an Open Recent menu to your application's File menu, although `NSDocumentController` will not attempt to remove any preexisting Open Recent menu item. The default implementation returns a value that is subject to change and may or may not be derived from a setting made by the user in System Preferences.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSDocumentController.h`

newDocument:

An action method invoked by the New menu command, this method creates a new `NSDocument` object and adds it to the list of such objects managed by the document controller.

```
- (IBAction)newDocument:(id)sender
```

Discussion
This method invokes `openUntitledDocumentAndDisplay:error:` (page 1024), as described in "Message Flow in the Document Architecture".

Availability
Available in Mac OS X v10.0 and later.

See Also
- openDocument: (page 1021)

Declared In
NSDocumentController.h

noteNewRecentDocument:

This method is called by NSDocument objects at appropriate times for managing the recent-documents list.

```
- (void)noteNewRecentDocument:(NSDocument *)aDocument
```

Discussion
This method constructs a URL and calls noteNewRecentDocumentURL: (page 1021). Subclasses might override this method to prevent certain documents or kinds of documents from getting into the list.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDocumentController.h

noteNewRecentDocumentURL:

This method should be called by applications not based on NSDocument when they open or save documents identified by the given URL.

```
- (void)noteNewRecentDocumentURL:(NSURL *)aURL
```

Discussion
NSDocument automatically calls this method when appropriate for NSDocument-based applications. Applications not based on NSDocument must also implement the application:openFile: method in the application delegate to handle requests from the Open Recent menu command. You can override this method in an NSDocument-based application to prevent certain kinds of documents from getting into the list (but you have to identify them by URL).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSDocumentController.h

openDocument:

An action method invoked by the Open menu command, it runs the modal Open panel and, based on the selected filenames, creates one or more NSDocument objects from the contents of the files.

```
- (IBAction)openDocument:(id)sender
```

Discussion
The method adds the newly created objects to the list of `NSDocument` objects managed by the document controller. This method invokes `openDocumentWithContentsOfURL:display:error:` (page 1023), which actually creates the `NSDocument` objects.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `fileNamesFromRunningOpenPanel` (page 1016)
- `newDocument:` (page 1020)

Related Sample Code
CIRAWFilterSample

Declared In
`NSDocumentController.h`

openDocumentWithContentsOfFile:display:

Returns a document object created from the contents of a given file and optionally displays it. (Deprecated in Mac OS X v10.4. Use `openDocumentWithContentsOfURL:display:error:` (page 1023) instead.)

```
- (id)openDocumentWithContentsOfFile:(NSString *)fileName display:(BOOL)flag
```

Discussion
Returns an `NSDocument` object created from the contents of the file *`fileName`* (an absolute path) and displays it if *`flag`* is `YES`. The returned object is not retained, but is added to the receiver's list of managed documents. Returns `nil` if the object could not be created, typically because *`fileName`* does not point to a valid file or because there is no `NSDocument` subclass for the document type (as indicated by the file extension or HFS file type). Even if *`flag`* is `YES`, the document is not displayed if `shouldCreateUI` (page 1029) returns `NO`. This method invokes `makeDocumentWithContentsOfFile:ofType:` (page 1017) to obtain the created NSDocument object. If you override this method, your implementation should be prepared to handle either `YES` or `NO`.

To handle an Open Documents Apple event, the Application Kit's built-in Apple event handling automatically invokes this method with the path to the file to open and a display argument.

Invoked with a display argument of `YES` instead of `NO` when a Print Documents Apple event is handled. This may have been handled differently in versions of Mac OS X prior to version 10.3.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- `openDocument:` (page 1021)
- `openUntitledDocumentOfType:display:` (page 1024)
- `setShouldCreateUI:` (page 1029)

Related Sample Code
ColorMatching
QTAudioContextInsert

QTAudioExtractionPanel
QTKitPlayer
QTMetadataEditor

Declared In
NSDocumentController.h

openDocumentWithContentsOfURL:display:

Returns a document object created from the contents of a given URL and optionally displays it. (Deprecated in Mac OS X v10.4. Use openDocumentWithContentsOfURL:display:error: (page 1023) instead.)

```
- (id)openDocumentWithContentsOfURL:(NSURL *)aURL display:(BOOL)flag
```

Discussion
Returns an NSDocument object created from the contents of *aURL* and displays it if *flag* is YES. The returned object is not retained, but is added to the receiver's list of managed documents. Returns nil if the object could not be created, typically because *aURL* does not point to a valid location or because there is no NSDocument subclass for the document type. Even if *flag* is YES, the document is not displayed if shouldCreateUI (page 1029) returns NO. This method invokes makeDocumentWithContentsOfURL:ofType: (page 1018) to obtain the created NSDocument object.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- openDocument: (page 1021)
- openUntitledDocumentOfType:display: (page 1024)
- setShouldCreateUI: (page 1029)

Declared In
NSDocumentController.h

openDocumentWithContentsOfURL:display:error:

Opens a document located by the given URL presents its user interface if requested, and returns the document if successful.

```
- (id)openDocumentWithContentsOfURL:(NSURL *)absoluteURL
    display:(BOOL)displayDocument error:(NSError **)outError
```

Discussion
If not successful, the method returns nil after setting *outError* to point to an NSError object that encapsulates the reason why the document could not be opened.

The default implementation of this method checks to see if the document is already open according to documentForURL: (page 1014), and if it is not open determines the type of the document, invokes makeDocumentWithContentsOfURL:ofType:error: (page 1019) to instantiate it, then invokes addDocument: (page 1009) to record its opening, and sends the document makeWindowControllers (page 961) and showWindows (page 987) messages if *displayDocument* is YES. If the document is already open it is just sent a showWindows (page 987) message if *displayDocument* is YES.

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method instead invokes `openDocumentWithContentsOfFile:display:` (page 1022), if it is overridden and the URL uses the `file:` scheme.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
Reviews

Declared In
`NSDocumentController.h`

openUntitledDocumentAndDisplay:error:

Creates a new untitled document, presents its user interface if *`displayDocument`* is `YES`, and returns the document if successful.

```
- (id)openUntitledDocumentAndDisplay:(BOOL)displayDocument error:(NSError **)outError
```

Discussion
If not successful, the method returns `nil` after setting *`outError`* to point to an NSError that encapsulates the reason why a new untitled document could not be created.

The default implementation of this method invokes `defaultType` (page 1011) to determine the type of new document to create, invokes `makeUntitledDocumentOfType:error:` (page 1020) to create it, then invokes `addDocument:` (page 1009) to record its opening. If *`displayDocument`* is `YES`, it then sends the new document `makeWindowControllers` (page 961) and `showWindows` (page 987) messages.

For backward binary compatibility with Mac OS X v10.3 and earlier, the default implementation of this method instead invokes `openUntitledDocumentOfType:display:` (page 1024) if it is overridden.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
FinalCutPro_AppleEvents
iSpend

Declared In
`NSDocumentController.h`

openUntitledDocumentOfType:display:

Returns a document object instantiated from the subclass of the given document type and optionally displays it. (Deprecated in Mac OS X v10.4. Use `openUntitledDocumentAndDisplay:error:` (page 1024) with `defaultType` (page 1011) instead.)

```
- (id)openUntitledDocumentOfType:(NSString *)docType display:(BOOL)display
```

Discussion
Returns an `NSDocument` object instantiated from the `NSDocument` subclass required by document type *`docType`* and displays it if *`flag`* is `YES`. The returned object is not retained, but is added to the receiver's list of managed documents. Returns `nil` if the object could not be created, typically because no `NSDocument` subclass could be found for *`docType`*. Even if *`flag`* is `YES`, the document is not displayed if `shouldCreateUI` (page 1029) returns `NO`. This method invokes `makeUntitledDocumentOfType:` (page 1019) to obtain the created `NSDocument` object.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- `newDocument:` (page 1020)
- `openDocumentWithContentsOfFile:display:` (page 1022)
- `setShouldCreateUI:` (page 1029)

Declared In
`NSDocumentController.h`

presentError:

Presents an error alert to the user as a modal panel.

```
- (BOOL)presentError:(NSError *)error
```

Discussion
Returns `YES` if error recovery was done, `NO` otherwise. This method does not return until the user dismisses the alert and, if the error has recovery options and a recovery delegate, the error's recovery delegate is sent an `attemptRecoveryFromError:optionIndex:` message.

The default `NSDocumentController` implementation of this method is equivalent to that of `NSResponder` while treating the application object as the next responder and forwarding error presentation messages to it. (The default `NSDocument` implementation of this method treats the shared `NSDocumentController` instance as the next responder and forwards these messages to it.) The default implementations of several `NSDocumentController` methods invoke this method.

The default implementation of this method invokes `willPresentError:` (page 1031) to give subclasses an opportunity to customize error presentation. You should not override this method but should instead override `willPresentError:` (page 1031).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `willPresentError:` (page 1031)
- `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 1026)

Declared In
`NSDocumentController.h`

presentError:modalForWindow:delegate:didPresentSelector:contextInfo:

Presents an error alert to the user as a modal panel.

```
- (void)presentError:(NSError *)error modalForWindow:(NSWindow *)window
    delegate:(id)delegate didPresentSelector:(SEL)didPresentSelector
    contextInfo:(void *)contextInfo
```

Discussion
When the user dismisses the alert and any recovery possible for the error and chosen by the user has been attempted, sends the message *didPresentSelector* to the specified *delegate*. The method selected by *didPresentSelector* must have the same signature as:

```
- (void)didPresentErrorWithRecovery:(BOOL)didRecover contextInfo:(void
*)contextInfo;
```

The default `NSDocumentController` implementation of this method is equivalent to that of `NSResponder` while treating the application object as the next responder and forwarding error presentation messages to it. (The default `NSDocument` implementation of this method treats the shared `NSDocumentController` instance as the next responder and forwards these messages to it.)

The default implementation of this method invokes `willPresentError:` (page 1031) to give subclasses an opportunity to customize error presentation. You should not override this method but should instead override `willPresentError:` (page 1031).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `willPresentError:` (page 1031)
- `presentError:` (page 1025)

Declared In
`NSDocumentController.h`

recentDocumentURLs

Returns the list of recent-document URLs.

```
- (NSArray *)recentDocumentURLs
```

Discussion
This method is not a good one to override since the internals of `NSDocumentController` do not generally use it.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDocumentController.h`

removeDocument:

Removes the given document from the list of open documents.

```
- (void)removeDocument:(NSDocument *)document
```

Discussion
A document will automatically call `removeDocument:` (page 1026) when it closes. This method is mostly provided for subclasses that want to know when documents close.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
iSpend

Declared In
`NSDocumentController.h`

reopenDocumentForURL:withContentsOfURL:error:

Reopens an autosaved document located by a URL, by reading the contents for the document from another URL, presents its user interface, and returns `YES` if successful.

```
- (BOOL)reopenDocumentForURL:(NSURL *)absoluteDocumentURL withContentsOfURL:(NSURL
    *)absoluteDocumentContentsURL error:(NSError **)outError
```

Discussion
The document is located by *absoluteDocumentURL* and the contents are read from *absoluteDocumentContentsURL*. If not successful, the method returns `NO` after setting *outError* to point to an `NSError` object that encapsulates the reason why the document could not be reopened.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSDocumentController.h`

reviewUnsavedDocumentsWithAlertTitle:cancellable:delegate:didReviewAllSelector:contextInfo:

Displays an alert dialog asking if the user wants to review unsaved documents (only if there are two or more unsaved documents), quit regardless of unsaved documents, or (if the choice is allowed) cancel the impending save operation.

```
- (void)reviewUnsavedDocumentsWithAlertTitle:(NSString *)title
    cancellable:(BOOL)cancellable delegate:(id)delegate
    didReviewAllSelector:(SEL)didReviewAllSelector contextInfo:(void *)contextInfo
```

Discussion
Assigns *delegate* to the panel. Invokes *didReviewAllSelector* with `YES` if quit without saving is chosen or if there are no dirty documents, and `NO` otherwise. If the user selects the "Review Unsaved" option, `closeAllDocumentsWithDelegate:didCloseAllSelector:contextInfo:` (page 1010) is invoked. This method is invoked when the user chooses the Quit menu command, and also when the computer power is being turned off. Note that *title* is ignored. Pass the *contextInfo* object with the callback.

The *didReviewAllSelector* callback method should have the following signature:

```
- (void)documentController:(NSDocumentController *)docController  didReviewAll:
 (BOOL)didReviewAll contextInfo:(void *)contextInfo
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDocumentController.h

runModalOpenPanel:forTypes:

Invokes the `NSOpenPanel` `runModalForTypes:` (page 1821) method, passing the *openPanel* object and the file *extensions* associated with a document type.

```
- (NSInteger)runModalOpenPanel:(NSOpenPanel *)openPanel forTypes:(NSArray
    *)extensions
```

Discussion
This method is invoked by the `URLsFromRunningOpenPanel` (page 1031) method. The *extensions* parameter may also contain encoded HFS file types as well as filename extensions.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDocumentController.h

saveAllDocuments:

As the action method invoked by the Save All command, saves all open documents of the application that need to be saved.

```
- (IBAction)saveAllDocuments:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `saveDocument:` (page 976) (`NSDocument`)

Declared In
NSDocumentController.h

setAutosavingDelay:

Sets the time interval in seconds for periodic autosaving.

```
- (void)setAutosavingDelay:(NSTimeInterval)autosavingDelay
```

Discussion
A value of 0 indicates that periodic autosaving should not be done at all. `NSDocumentController` uses this number as the amount of time to wait between detecting that a document has unautosaved changes and sending the document an `autosaveDocumentWithDelegate:didAutosaveSelector:contextInfo:` (page 941) message. The default value is 0.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- autosavingDelay` (page 1009)

Related Sample Code
Sketch+Accessibility

Declared In
`NSDocumentController.h`

setShouldCreateUI:

Sets whether the window controllers of a document should be created when the document is created. (Deprecated in Mac OS X v10.4. Use the display parameter of `openUntitledDocumentAndDisplay:error:` (page 1024) or `openDocumentWithContentsOfURL:display:error:` (page 1023) instead.)

```
- (void)setShouldCreateUI:(BOOL)flag
```

Discussion
Sets whether the window controllers (`NSWindowController` instances) of a document should be created when the document is created. When a window controller is created, it loads the nib file containing the window it manages. Often *`flag`* is set to `NO` for scripting or searching operations involving the document's data.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
`- shouldCreateUI` (page 1029)

Declared In
`NSDocumentController.h`

shouldCreateUI

Returns a Boolean value that indicates whether the window controllers of a document should be created when the document is created. (Deprecated in Mac OS X v10.4. Use the *`display`* parameter of `openUntitledDocumentAndDisplay:error:` (page 1024) or `openDocumentWithContentsOfURL:display:error:` (page 1023) instead.)

```
- (BOOL)shouldCreateUI
```

Return Value
A Boolean value that indicates whether the window controllers (`NSWindowController` instances) of a document should be created when the document is created.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
`- setShouldCreateUI:` (page 1029)

Declared In
`NSDocumentController.h`

typeForContentsOfURL:error:

Returns, for a specified URL, the name of the document type that should be used when opening the document at that location, if successful.

```
- (NSString *)typeForContentsOfURL:(NSURL *)inAbsoluteURL error:(NSError **)outError
```

Discussion
The URL is represented by *absoluteURL*. If not successful, the method returns `nil` after setting *outError* to point to an `NSError` object that encapsulates the reason why the document type could not be determined, or the fact that the document type is unrecognized.

You can override this method to customize type determination for documents being opened.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSDocumentController.h`

typeFromFileExtension:

Returns the document type associated with files having extension *fileExtensionOrHFSFileType*. (Deprecated in Mac OS X v10.5. Use `typeForContentsOfURL:error:` (page 1030) instead.)

```
- (NSString *)typeFromFileExtension:(NSString *)fileExtensionOrHFSFileType
```

Discussion
The *fileExtensionOrHFSFileType* parameter may also be an encoded HFS file type, as well as a filename extension.

This method only works when passed a file name extension used in a `CFBundleDocumentTypes` entry that does not have an `LSItemContentTypes` subentry.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

See Also
- `displayNameForType:` (page 1012)

Declared In
`NSDocumentController.h`

URLsFromRunningOpenPanel

Creates an `NSOpenPanel` instance and initializes it appropriately.

```
- (NSArray *)URLsFromRunningOpenPanel
```

Discussion
This method uses `runModalOpenPanel:forTypes:` (page 1028) to run the open panel. Returns the chosen files as an array of URLs. Returns `nil` if the user cancels the Open panel or makes no selection.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDocumentController.h`

validateUserInterfaceItem:

Returns a Boolean value that indicates whether a given user interface item should be enabled.

```
- (BOOL)validateUserInterfaceItem:(id < NSValidatedUserInterfaceItem >)anItem
```

Parameters

anItem

The user interface item to validate. You can send *anItem* the `action` (page 3925) and `tag` (page 3926) messages.

Return Value
`YES` if *anItem* should be enabled, otherwise `NO`.

Discussion
Subclasses can override this method to perform additional validations. Subclasses should call `super` in their implementation for items they don't handle themselves.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDocumentController.h`

willPresentError:

Called when the receiver is about to present an error, returns the error that should actually be presented.

```
- (NSError *)willPresentError:(NSError *)error
```

Discussion
The default implementation of this method merely returns the passed-in error. The returned error may simply be forwarded to the application object.

You can override this method to customize the presentation of errors by examining the passed-in error and, for example, returning more specific information. When you override this method always check the `NSError` object's domain and code to discriminate between errors whose presentation you want to customize and those you don't. For errors you don't want to customize, call the superclass implementation, passing the original error.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `presentError:` (page 1025)
- `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 1026)

Declared In
`NSDocumentController.h`

NSDrawer Class Reference

Inherits from	NSResponder : NSObject
Conforms to	NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSDrawer.h
Companion guide	Drawers
Related sample code	CocoaAUHost DrawerMadness PDFKitLinker2 QTAudioExtractionPanel

Overview

An `NSDrawer` object is a user interface element that contains and displays view objects including `NSTextView`, `NSScrollView`, `NSBrowser`, and other classes that inherit from `NSView`. A drawer is associated with a window, called its parent, and can appear only while its parent is visible onscreen. A drawer cannot be moved or ordered independently of a window, but is instead attached to one edge of its parent and moves along with it.

Tasks

Creating Drawers

`- initWithContentSize:preferredEdge:` (page 1037)
Creates a new drawer with the given size on the specified edge of the parent window.

`- delegate` (page 1037)
Returns the receiver's delegate.

`- setDelegate:` (page 1042)
Sets the receiver's delegate.

Opening and Closing Drawers

- `close` (page 1035)
  If the receiver is open, this method closes it.
- `close:` (page 1036)
  An action method to close the receiver.
- `open` (page 1039)
  If the receiver is closed, this method opens it.
- `open:` (page 1039)
  An action method to open the drawer.
- `openOnEdge:` (page 1040)
  Causes the receiver to open on the specified edge of the parent window.
- `toggle:` (page 1045)
  Toggles the drawer open or closed.
- `state` (page 1044)
  Returns the state of the receiver.

Managing Drawer Size

- `contentSize` (page 1036)
  Returns the size of the receiver's content area.
- `leadingOffset` (page 1038)
  Returns the receiver's leading offset.
- `maxContentSize` (page 1038)
  Returns the maximum allowed size of the receiver's content area.
- `minContentSize` (page 1039)
  Returns the minimum allowed size of the receiver's content area.
- `setContentSize:` (page 1041)
  Sets the size of the receiver's content area.
- `setLeadingOffset:` (page 1042)
  Sets the receiver's leading offset.
- `setMaxContentSize:` (page 1042)
  Specifies the maximum size of the receiver's content area.
- `setMinContentSize:` (page 1043)
  Specifies the minimum size of the receiver's content area.
- `setTrailingOffset:` (page 1044)
  Sets the receiver's trailing offset.
- `trailingOffset` (page 1045)
  Returns the receiver's trailing offset.

Managing Drawer Edges

- `edge` (page 1037)
  Returns the edge of the window that the receiver is connected to.
- `preferredEdge` (page 1040)
  Returns the receiver's preferred, or default, edge.
- `setPreferredEdge:` (page 1044)
  Sets the receiver's preferred, or default, edge.

Managing Drawer Views

- `contentView` (page 1036)
  Returns the receiver's content view.
- `parentWindow` (page 1040)
  Returns the receiver's parent window.
- `setContentView:` (page 1041)
  Sets the receiver's content view.
- `setParentWindow:` (page 1043)
  Sets the receiver's parent window.

Instance Methods

close

If the receiver is open, this method closes it.

```
- (void)close
```

Discussion
Calling *close* on a closed drawer does nothing. You can get the state of a drawer by sending it `state` (page 1044).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `open` (page 1039)

Related Sample Code
DrawerMadness

Declared In
`NSDrawer.h`

close:

An action method to close the receiver.

```
- (void)close:(id)sender
```

Parameters

sender

A user interface element, such as a button or menu item, that invokes the action method.

Discussion

This method is an action method and likely would not be invoked programatically. Rather, it is an action that is commonly connected in Interface Builder.

Availability

Available in Mac OS X v10.0 and later.

See Also

- open: (page 1039)

Declared In

NSDrawer.h

contentSize

Returns the size of the receiver's content area.

```
- (NSSize)contentSize
```

Return Value

The size of the receiver's content area.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setContentSize: (page 1041)
- setMaxContentSize: (page 1042)
- setMinContentSize: (page 1043)

Declared In

NSDrawer.h

contentView

Returns the receiver's content view.

```
- (NSView *)contentView
```

Return Value

The receiver's content view.

Availability

Available in Mac OS X v10.0 and later.

See Also
- setContentView: (page 1041)

Declared In
NSDrawer.h

delegate

Returns the receiver's delegate.

```
- (id < NSDrawerDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setDelegate: (page 1042)

Declared In
NSDrawer.h

edge

Returns the edge of the window that the receiver is connected to.

```
- (NSRectEdge)edge
```

Return Value
The edge of the parent window at which the drawer is attached. See "Constants" (page 1046) for a list of edge constants and locations.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDrawer.h

initWithContentSize:preferredEdge:

Creates a new drawer with the given size on the specified edge of the parent window.

```
- (id)initWithContentSize:(NSSize)contentSize preferredEdge:(NSRectEdge)edge
```

Parameters

contentSize
The size of the new drawer.

edge
The edge to which to attach the new drawer.

Discussion
You must specify the parent window and content view of the drawer using the methods in this class. When you create a drawer in Interface Builder, this constructor is invoked. The NSDrawer Inspector in Interface Builder allows you to set the edge, and you can specify the size by changing the content view in Interface Builder.

See Positioning and Sizing a Drawer for additional detail on content size and drawer positioning.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DrawerMadness

Declared In
`NSDrawer.h`

leadingOffset

Returns the receiver's leading offset.

```
- (CGFloat)leadingOffset
```

Return Value
The receiver's leading offset. This is the distance from the top or left edge of the parent window to the drawer.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setLeadingOffset:` (page 1042)

Declared In
`NSDrawer.h`

maxContentSize

Returns the maximum allowed size of the receiver's content area.

```
- (NSSize)maxContentSize
```

Return Value
The maximum size of the receiver's content area. This is useful for determining if an opened drawer would fit onscreen given the current window position.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMaxContentSize:` (page 1042)

Declared In
`NSDrawer.h`

minContentSize

Returns the minimum allowed size of the receiver's content area.

```
- (NSSize)minContentSize
```

Return Value
The minimum size of the receiver's content area.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMinContentSize:` (page 1043)

Declared In
`NSDrawer.h`

open

If the receiver is closed, this method opens it.

```
- (void)open
```

Discussion
Calling *open* on an open drawer does nothing. You can get the state of a drawer by sending it `state` (page 1044). If an edge is not specified, an attempt will be made to choose an edge based on the space available to display the drawer onscreen. If you need to ensure that a drawer opens on a particular edge, use `openOnEdge:` (page 1040).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- close` (page 1035)

Declared In
`NSDrawer.h`

open:

An action method to open the drawer.

```
- (void)open:(id)sender
```

Parameters
sender
A user interface element, such as a button or menu item, that invokes the action method.

Discussion
This method is an action method and likely would not be invoked programatically. Rather, it is an action that is commonly connected in Interface Builder.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `close:` (page 1036)

Declared In
`NSDrawer.h`

openOnEdge:

Causes the receiver to open on the specified edge of the parent window.

```
- (void)openOnEdge:(NSRectEdge)edge
```

Parameters

edge

The edge of the parent window on which to open the receiver. See "Constants" (page 1046) for a list of edge constants and locations.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DrawerMadness

Declared In
`NSDrawer.h`

parentWindow

Returns the receiver's parent window.

```
- (NSWindow *)parentWindow
```

Return Value
The receiver's parent window. By definition, a drawer can appear onscreen only if it has a parent.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setParentWindow:` (page 1043)

Declared In
`NSDrawer.h`

preferredEdge

Returns the receiver's preferred, or default, edge.

```
- (NSRectEdge)preferredEdge
```

Return Value
The receiver's preferred edge. If a drawer is told to open and an edge is not specified at that time, it opens on its preferred edge. When you a create a drawer with Interface Builder, the preferred edge is set to the left by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPreferredEdge:` (page 1044)

Declared In
`NSDrawer.h`

setContentSize:

Sets the size of the receiver's content area.

```
- (void)setContentSize:(NSSize)size
```

Parameters

size
The new size of the receiver's content area. See Positioning and Sizing a Drawer for additional detail.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `contentSize` (page 1036)
- `setMaxContentSize:` (page 1042)
- `setMinContentSize:` (page 1043)

Declared In
`NSDrawer.h`

setContentView:

Sets the receiver's content view.

```
- (void)setContentView:(NSView *)aView
```

Parameters

aView
The content view of the receiver. Rather than connect a drawer to its content view in Interface Builder, you can specify it programatically with this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `contentView` (page 1036)

Declared In
NSDrawer.h

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSDrawerDelegate >)anObject
```

Parameters

anObject

The object to assign as the receiver's delegate. The object must conform to the *NSDrawerDelegate Protocol Reference*.

Discussion
You may find it useful to associate a delegate with a drawer, especially since drawers do not open and close instantly. A drawer's delegate can better regulate drawer behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
- delegate (page 1037)

Declared In
NSDrawer.h

setLeadingOffset:

Sets the receiver's leading offset.

```
- (void)setLeadingOffset:(CGFloat)offset
```

Parameters

offset

The leading offset of the receiver. This is the distance from the top or left edge of the parent window to the drawer. See Positioning and Sizing a Drawer for additional detail.

Availability
Available in Mac OS X v10.0 and later.

See Also
- leadingOffset (page 1038)
- setTrailingOffset: (page 1044)

Declared In
NSDrawer.h

setMaxContentSize:

Specifies the maximum size of the receiver's content area.

```
- (void)setMaxContentSize:(NSSize)size
```

Parameters

size

The new maximum size of the receiver's content area. See Positioning and Sizing a Drawer for additional detail.

Availability
Available in Mac OS X v10.0 and later.

See Also
- maxContentSize (page 1038)

Declared In
NSDrawer.h

setMinContentSize:

Specifies the minimum size of the receiver's content area.

```
- (void)setMinContentSize:(NSSize)size
```

Parameters

size

The new minimum size of the receiver's content area. See Positioning and Sizing a Drawer for additional detail.

Availability
Available in Mac OS X v10.0 and later.

See Also
- minContentSize (page 1039)

Declared In
NSDrawer.h

setParentWindow:

Sets the receiver's parent window.

```
- (void)setParentWindow:(NSWindow *)parent
```

Parameters

parent

The parent window of the receiver. Every drawer must be associated with a parent window for a drawer to appear onscreen. If this argument is nil, the drawer is removed from its parent.

Discussion
Changes in a drawer's parent window do not take place while the drawer is onscreen; they are delayed until the drawer next closes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- parentWindow (page 1040)

Declared In
NSDrawer.h

setPreferredEdge:

Sets the receiver's preferred, or default, edge.

```
- (void)setPreferredEdge:(NSRectEdge)preferredEdge
```

Parameters

preferredEdge

The edge on which the receiver should open by default. A drawer can be told to open on a specific edge (page 1037); if an edge is not specified, however, it opens on the preferred edge.

Availability
Available in Mac OS X v10.0 and later.

See Also
- preferredEdge (page 1040)

Declared In
NSDrawer.h

setTrailingOffset:

Sets the receiver's trailing offset.

```
- (void)setTrailingOffset:(CGFloat)offset
```

Parameters

offset

The receiver's trailing offset. This is the distance to the right or bottom edge of the drawer from the right or bottom edge of the parent window. See Positioning and Sizing a Drawer for additional detail.

Availability
Available in Mac OS X v10.0 and later.

See Also
- leadingOffset (page 1038)

Declared In
NSDrawer.h

state

Returns the state of the receiver.

```
- (NSInteger)state
```

Return Value
The drawer's state. Refer to `NSDrawerState` (page 1046) for a list of possible values.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DrawerMadness

Declared In
`NSDrawer.h`

toggle:

Toggles the drawer open or closed.

```
- (void)toggle:(id)sender
```

Parameters

sender
The sender of the message.

Discussion
If the receiver is closed, or in the process of either opening or closing, it is opened. Otherwise, the drawer is closed.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTAudioExtractionPanel

Declared In
`NSDrawer.h`

trailingOffset

Returns the receiver's trailing offset.

```
- (CGFloat)trailingOffset
```

Return Value
The receiver's trailing offset. This is the distance to the right or bottom edge of the drawer from the right or bottom edge of the parent window.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTrailingOffset:` (page 1044)

Declared In
`NSDrawer.h`

Constants

NSDrawerState

These constants specify the possible states of a drawer.

```
typedef enum _NSDrawerState {
    NSDrawerClosedState  = 0,
    NSDrawerOpeningState = 1,
    NSDrawerOpenState    = 2,
    NSDrawerClosingState = 3
} NSDrawerState;
```

Constants

`NSDrawerClosedState`
The drawer is closed (not visible onscreen).
Available in Mac OS X v10.0 and later.
Declared in `NSDrawer.h`.

`NSDrawerOpeningState`
The drawer is in the process of opening.
Available in Mac OS X v10.0 and later.
Declared in `NSDrawer.h`.

`NSDrawerOpenState`
The drawer is open (visible onscreen).
Available in Mac OS X v10.0 and later.
Declared in `NSDrawer.h`.

`NSDrawerClosingState`
The drawer is in the process of closing.
Available in Mac OS X v10.0 and later.
Declared in `NSDrawer.h`.

Discussion
These constants are returned by `state` (page 1044).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDrawer.h`

Notifications

NSDrawerDidCloseNotification

Posted whenever the drawer is closed.

The notification object is the `NSDrawer` object that closed. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDrawer.h`

NSDrawerDidOpenNotification

Posted whenever the drawer is opened.

The notification object is the `NSDrawer` object that opened. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDrawer.h`

NSDrawerWillCloseNotification

Posted whenever the drawer is about to close.

The notification object is the `NSDrawer`object about to close. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDrawer.h`

NSDrawerWillOpenNotification

Posted whenever the drawer is about to open.

The notification object is the `NSDrawer` object about to open. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDrawer.h`

NSEPSImageRep Class Reference

Inherits from	NSImageRep : NSObject
Conforms to	NSCoding (NSImageRep) NSCopying (NSImageRep) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSEPSImageRep.h
Companion guide	Cocoa Drawing Guide

Overview

An `NSEPSImageRep` object can render an image from encapsulated PostScript (EPS) code.

Tasks

Creating an NSEPSImageRep

`+ imageRepWithData:` (page 1050)
Creates and returns an `NSEPSImageRep` object initialized with the specified EPS data.

`- initWithData:` (page 1051)
Returns an `NSEPSImageRep` object initialized with the specified EPS data.

Getting Image Data

`- boundingBox` (page 1050)
Returns the rectangle that bounds the receiver.

`- EPSRepresentation` (page 1051)
Returns the EPS representation of the receiver.

Drawing the Image

- prepareGState (page 1051)
  Implemented by subclasses to configure the graphics state prior to drawing.

Class Methods

imageRepWithData:

Creates and returns an NSEPSImageRep object initialized with the specified EPS data.

```
+ (id)imageRepWithData:(NSData *)epsData
```

Parameters

epsData
The EPS data representing the desired image.

Return Value
A new, initialized NSEPSImageRep object or nil if the object could not be initialized.

Discussion
The size of the receiver is set using the bounding box information specified in the EPS header comments.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSEPSImageRep.h

Instance Methods

boundingBox

Returns the rectangle that bounds the receiver.

```
- (NSRect)boundingBox
```

Return Value
The bounding box of the receiver. This rectangle is obtained from the "%%BoundingBox:" comment in the EPS header when the NSEPSImageRep object is initialized.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ imageRepWithData: (page 1050)
- initWithData: (page 1051)

Declared In
NSEPSImageRep.h

EPSRepresentation

Returns the EPS representation of the receiver.

```
- (NSData *)EPSRepresentation
```

Return Value
A data object containing the EPS data for the image.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSEPSImageRep.h

initWithData:

Returns an `NSEPSImageRep` object initialized with the specified EPS data.

```
- (id)initWithData:(NSData *)epsData
```

Parameters
epsData
The EPS data representing the desired image.

Return Value
The initialized `NSEPSImageRep` object or `nil` if the object could not be initialized

Discussion
The size of the receiver is set using the bounding box information specified in the EPS header comments.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSEPSImageRep.h

prepareGState

Implemented by subclasses to configure the graphics state prior to drawing.

```
- (void)prepareGState
```

Discussion
The `draw` (page 1415) method of `NSEPSImageRep` sends this message to itself just before rendering the EPS code. The default implementation of this method does nothing. You can override it in your subclass to prepare the graphics state as needed.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSEPSImageRep.h`

NSEvent Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSEvent.h
Companion guide	Cocoa Event-Handling Guide
Related sample code	Cocoa OpenGL DragItemAround GLUT LightTable Sketch-112

Overview

An `NSEvent` object, or simply an event, contains information about an input action such as a mouse click or a key down. The Application Kit associates each such user action with a window, reporting the event to the application that created the window. The `NSEvent` object contains pertinent information about each event, such as where the cursor was located or which character was typed. As the application receives events, it temporarily places them in a buffer called the event queue. When the application is ready to process an event, it takes one from the queue.

Beginning with Mac OS X version 10.4, `NSEvent` objects can represent tablet-pointing and tablet-proximity events. A tablet-proximity event is generated when a pointing device enters or leaves proximity of its tablet; such event objects have a type of `NSTypeProximity` or a mouse subtype of `NSTabletProximityEventSubtype`. A tablet-pointing event is generated when a pointing device changes state, such as location, pressure, or tilt; such event objects have a type of `NSTypePoint` or a mouse subtype of `NSTabletPointEventSubtype`. The Application Kit reports all pure tablet events to responder objects through the `NSResponder` methods `tabletPoint:` (page 2199) and `tabletProximity:` (page 2200). Mouse events can also contain tablet data (as event subtypes), so you can handle these events by overriding the `NSResponder` methods `mouseDown:` (page 2164), `mouseDragged:` (page 2164), and `mouseUp:` (page 2166).

Support for touch and gesture events masks have been added to `NSEvent` in Mac OS X v10.6. Magnify (pinch), swipe, and rotate masks are supported, as are more generic gesture masks. Using `touchesMatchingPhase:inView:` (page 1086) method a view can get all of the touch events associated with a gesture without overriding the individual touch responder methods defined in `NSResponder`.

Mac OS X v10.6 adds the ability to create application event monitors that call block object handlers for certain event types that are sent through the `NSApplication sendEvent:` (page 167) method. You can create a local monitor that will be informed of the events in your application and allow you to modify or cancel them. You can also create a global event monitor that allows you to monitor events in other applications, although you are unable to alter those events. See "Monitoring Application Events" (page 1058) for more information.

Tasks

Creating Events

`+ keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode:` (page 1063)

Returns a new `NSEvent` object describing a key event.

`+ mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure:` (page 1066)

Returns a new NSEvent object describing a mouse-down, -up, -moved, or -dragged event.

`+ enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:` (page 1061)

Returns a new `NSEvent` object describing a tracking-rectangle or cursor-update event.

`+ otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2:` (page 1067)

Returns a new NSEvent object describing a custom event.

`+ eventWithEventRef:` (page 1063)

Creates an event object that is based on a Carbon type of event.

`+ eventWithCGEvent:` (page 1062)

Creates and returns an event object that is based on a Core Graphics type of event.

Getting General Event Information

`- context` (page 1075)

Returns the display graphics context of the receiver.

`- locationInWindow` (page 1080)

Returns the receiver's location in the base coordinate system of the associated window.

`- modifierFlags` (page 1082)

Returns an integer bit field indicating the modifier keys in effect for the receiver.

`- timestamp` (page 1086)

Returns the time the receiver occurred in seconds since system startup.

`- type` (page 1088)

Returns the type of the receiving event.

- `window` (page 1090)
  Returns the window object associated with the receiver.
- `windowNumber` (page 1091)
  Returns the identifier for the window device associated with the receiver.
- `eventRef` (page 1079)
  Returns the Carbon type associated with the receiver for representing an event.
- `CGEvent` (page 1073)
  Returns a Core Graphics event object corresponding to the receiver.

Getting Key Event Information

+ `modifierFlags` (page 1065)
  Returns the currently pressed modifier flags.
+ `keyRepeatDelay` (page 1065)
  Returns the length of time a key must be held down in order to generate the first key repeat event.
+ `keyRepeatInterval` (page 1065)
  Returns the length between subsequent key repeat events being posted.
- `characters` (page 1074)
  Returns the characters associated with the receiving key-up or key-down event.
- `charactersIgnoringModifiers` (page 1074)
  Returns the characters generated by the receiving key event as if no modifier key (except for Shift) applies.
- `isARepeat` (page 1079)
  Returns `YES` if the receiving key event is a repeat caused by the user holding the key down, `NO` if the key event is new.
- `keyCode` (page 1080)
  Returns the virtual key code for the keyboard key associated with the receiving key event.

Getting Mouse Event Information

+ `pressedMouseButtons` (page 1068)
  Returns the indices of the currently depressed mouse buttons.
+ `doubleClickInterval` (page 1061)
  Returns the time, in seconds, in which a second mouse click must occur in order to be considered a double click.
+ `mouseLocation` (page 1067)
  Reports the current mouse position in screen coordinates.
- `buttonNumber` (page 1073)
  Returns the button number for the mouse button that generated an `NSOtherMouse...` event.
- `clickCount` (page 1075)
  Returns the number of mouse clicks associated with the receiver, which represents a mouse-down or mouse-up event.

- `pressure` (page 1083)
  Returns a value from 0.0 through 1.0 indicating the pressure applied to the input device (used for appropriate devices).
+ `setMouseCoalescingEnabled:` (page 1069)
  Controls whether mouse-movement event coalescing is enabled.
+ `isMouseCoalescingEnabled` (page 1063)
  Indicates whether mouse-movement event coalescing is enabled.

Getting Mouse-Tracking Event Information

- `eventNumber` (page 1078)
  Returns the counter value of the latest mouse or tracking-rectangle event object; every system-generated mouse and tracking-rectangle event increments this counter.
- `trackingNumber` (page 1087)
  Returns the identifier of a mouse-tracking event.
- `trackingArea` (page 1087)
  Returns the `NSTrackingArea` object that generated the event represented by the receiver.
- `userData` (page 1089)
  Returns data associated with a mouse-tracking event,

Getting Custom Event Information

- `data1` (page 1076)
  Returns additional data associated with the receiver.
- `data2` (page 1076)
  Returns additional data associated with the receiver.
- `subtype` (page 1084)
  Returns the subtype of the receiving event object.

Getting Scroll Wheel Event Information

- `deltaX` (page 1077)
  Returns the x-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.
- `deltaY` (page 1077)
  Returns the y-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.
- `deltaZ` (page 1077)
  Returns the z-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.

Getting Tablet Proximity Information

- `capabilityMask` (page 1073)
  Returns a mask whose set bits indicate the capabilities of the tablet device that generated the event represented by the receiver.

- `deviceID` (page 1078)
  Returns a special identifier that is used to match tablet-pointer events with the tablet-proximity event represented by the receiver.
- `isEnteringProximity` (page 1080)
  Returns `YES` to indicate that a pointing device is entering the proximity of its tablet and `NO` when it is leaving it.
- `pointingDeviceID` (page 1082)
  Returns the index of the pointing device currently in proximity with the tablet.
- `pointingDeviceSerialNumber` (page 1082)
  Returns the vendor-assigned serial number of a pointing device of a certain type.
- `pointingDeviceType` (page 1083)
  Returns a `NSPointingDeviceType` constant indicating the kind of pointing device associated with the receiver.
- `systemTabletID` (page 1085)
  Returns the index of the tablet device connected to the system.
- `tabletID` (page 1085)
  Returns the USB model identifier of the tablet device associated with the receiver.
- `uniqueID` (page 1088)
  Returns the unique identifier of the pointing device that generated the event represented by the receiver.
- `vendorID` (page 1090)
  Returns the vendor identifier of the tablet associated with the receiver.
- `vendorPointingDeviceType` (page 1090)
  Returns a coded bit field whose set bits indicate the type of pointing device (within a vendor selection) associated with the receiver.

Getting Tablet Pointing Information

- `absoluteX` (page 1071)
  Reports the absolute x coordinate of a pointing device on its tablet at full tablet resolution.
- `absoluteY` (page 1071)
  Reports the absolute y coordinate of a pointing device on its tablet at full tablet resolution.
- `absoluteZ` (page 1072)
  Reports the absolute z coordinate of pointing device on its tablet at full tablet resolution.
- `buttonMask` (page 1072)
  Returns a bit mask identifying the buttons pressed when the tablet event represented by the receiver was generated.
- `rotation` (page 1084)
  Returns the rotation in degrees of the tablet pointing device associated with the receiver.
- `tangentialPressure` (page 1085)
  Reports the tangential pressure on the device that generated the event represented by the receiver.
- `tilt` (page 1086)
  Reports the scaled tilt values of the pointing device that generated the event represented by the receiver.

- `vendorDefined` (page 1089)
  Returns an array of three vendor-defined `NSNumber` objects associated with the pointing-type event represented by the receiver.

Requesting and Stopping Periodic Events

+ `startPeriodicEventsAfterDelay:withPeriod:` (page 1070)
  Begins generating periodic events for the current thread.

+ `stopPeriodicEvents` (page 1070)
  Stops generating periodic events for the current thread and discards any periodic events remaining in the queue.

Getting Touch and Gesture Information

- `magnification` (page 1081)
  Returns the change in magnification.

- `touchesMatchingPhase:inView:` (page 1086)
  Returns all the `NSTouch` objects associated with a specific phase.

Monitoring Application Events

+ `addGlobalMonitorForEventsMatchingMask:handler:` (page 1058)
  Installs an event monitor that receives copies of events posted to other applications.

+ `addLocalMonitorForEventsMatchingMask:handler:` (page 1060)
  Installs an event monitor that receives copies of events posted to this application before they are dispatched.

+ `removeMonitor:` (page 1069)
  Remove the specified event monitor.

Class Methods

addGlobalMonitorForEventsMatchingMask:handler:

Installs an event monitor that receives copies of events posted to other applications.

```
+ (id)addGlobalMonitorForEventsMatchingMask:(NSEventMask)mask handler:(void
    (^)(NSEvent*))block
```

Parameters

mask
  An event mask specifying which events you wish to monitor. See `Masks for event types` (page 1096) for possible values.

block

The event handler block object. It is passed the event to monitor. You are unable to change the event, merely observe it.

Return Value
An event handler object.

Discussion
Events are delivered asynchronously to your app and you can only observe the event; you cannot modify or otherwise prevent the event from being delivered to its original target application.

Key-related events may only be monitored if accessibility is enabled or if your application is trusted for accessibility access (see `AXIsProcessTrusted`).

Note that your handler will not be called for events that are sent to your own application.

Special Considerations

In Mac OS X v 10.6, event monitors are only able to monitor the following event types:

- `NSFlagsChanged` (page 1095)
- `NSLeftMouseDragged` (page 1094)
- `NSRightMouseDragged` (page 1094)
- `NSOtherMouseDragged` (page 1094)
- `NSLeftMouseUp` (page 1093)
- `NSRightMouseUp` (page 1093)
- `NSOtherMouseUp` (page 1094)
- `NSLeftMouseDown` (page 1093)
- `NSRightMouseDown` (page 1093)
- `NSOtherMouseDown` (page 1094)
- `NSMouseMoved` (page 1094)
- `NSFlagsChanged` (page 1095)
- `NSScrollWheel` (page 1095)
- `NSTabletPoint` (page 1095)
- `NSTabletProximity` (page 1095)
- `NSKeyDown` (page 1094) (Key repeats are determined by sending the event an `isARepeat` (page 1079) message.)

Availability
Available in Mac OS X v10.6 and later.

See Also
`+ addLocalMonitorForEventsMatchingMask:handler:` (page 1060)
`+ removeMonitor:` (page 1069)

Declared In
`NSEvent.h`

addLocalMonitorForEventsMatchingMask:handler:

Installs an event monitor that receives copies of events posted to this application before they are dispatched.

```
+ (id)addLocalMonitorForEventsMatchingMask:(NSEventMask)mask handler:(NSEvent*
    (^)(NSEvent*))block
```

Parameters

mask

An event mask specifying which events you wish to monitor. See `Masks for event types` (page 1096) for possible values.

block

The event handler block object. It is passed the event to monitor. You can return the event unmodified, create and return a new NSEvent object, or return nil to stop the dispatching of the event.

Return Value

An event handler object.

Discussion

Your handler will not be called for events that are consumed by nested event-tracking loops such as control tracking, menu tracking, or window dragging; only events that are dispatched through the applications `sendEvent:` (page 167) method will be passed to your handler.

Special Considerations

In Mac OS X v 10.6, event monitors are only able to monitor the following event types:

- `NSFlagsChanged` (page 1095)
- `NSLeftMouseDragged` (page 1094)
- `NSRightMouseDragged` (page 1094)
- `NSOtherMouseDragged` (page 1094)
- `NSLeftMouseUp` (page 1093)
- `NSRightMouseUp` (page 1093)
- `NSOtherMouseUp` (page 1094)
- `NSLeftMouseDown` (page 1093)
- `NSRightMouseDown` (page 1093)
- `NSOtherMouseDown` (page 1094)
- `NSMouseMoved` (page 1094)
- `NSFlagsChanged` (page 1095)
- `NSScrollWheel` (page 1095)
- `NSTabletPoint` (page 1095)
- `NSTabletProximity` (page 1095)
- `NSKeyDown` (page 1094) (Key repeats are determined by sending the event an `isARepeat` (page 1079) message.)

Availability

Available in Mac OS X v10.6 and later.

See Also
`+ addGlobalMonitorForEventsMatchingMask:handler:` (page 1058)
`+ removeMonitor:` (page 1069)

Related Sample Code
AnimatedTableView

Declared In
`NSEvent.h`

doubleClickInterval

Returns the time, in seconds, in which a second mouse click must occur in order to be considered a double click.

```
+ (NSTimeInterval)doubleClickInterval
```

Return Value
The double-click time interval, in seconds.

Discussion
This is a system setting, overriding this method will have no effect.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSEvent.h`

enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:

Returns a new `NSEvent` object describing a tracking-rectangle or cursor-update event.

```
+ (NSEvent *)enterExitEventWithType:(NSEventType)type location:(NSPoint)location
    modifierFlags:(NSUInteger)flags timestamp:(NSTimeInterval)time
    windowNumber:(NSInteger)windowNumber context:(NSGraphicsContext *)context
    eventNumber:(NSInteger)eventNumber trackingNumber:(NSInteger)trackingNumber
    userData:(void *)userData
```

Parameters

type
One of the following event-type constants: `NSMouseEntered`, `NSMouseExited`, `NSCursorUpdate`. If the specified constant is not one of these, an `NSInternalInconsistencyException` is raised

location
The cursor location in the base coordinate system of the window specified by *windowNum*.

flags
An integer bit field containing any of the modifier key masks described in "Constants" (page 1091), combined using the C bitwise OR operator.

time
The time the event occurred in seconds since system startup.

windowNum
: An integer that identifies the window device associated with the event, which is associated with the `NSWindow` that will receive the event.

context
: The display graphics context of the event.

eventNumber
: An identifier for the new event. It's normally taken from a counter for mouse events, which continually increases as the application runs.

trackingNumber
: A number that identifies the tracking rectangle. This identifier is the same as that returned by the `NSView` method `addTrackingRect:owner:userData:assumeInside:` (page 3142).

userData
: Data arbitrarily associated with the tracking rectangle when it was set up using the `NSView` method `addTrackingRect:owner:userData:assumeInside:` (page 3142).

Return Value
The created `NSEvent` object or `nil` if the object could not be created.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `eventNumber` (page 1078)
- `trackingNumber` (page 1087)
- `userData` (page 1089)

Declared In
`NSEvent.h`

eventWithCGEvent:

Creates and returns an event object that is based on a Core Graphics type of event.

```
+ (NSEvent *)eventWithCGEvent:(CGEventRef)cgEvent
```

Parameters

cgEvent
: A `CGEventRef` opaque type that represents an event.

Return Value
An autoreleased `NSEvent` object that is equivalent to *cgEvent*.

Discussion
The returned object retains the `CGEventRef` object (`cgEvent`) until it (the Objective-C object) is freed—it then releases the `CGEventRef` object. If no Cocoa event corresponds to the `CGEventRef` object, this method returns `nil`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `CGEvent` (page 1073)

Declared In
NSEvent.h

eventWithEventRef:

Creates an event object that is based on a Carbon type of event.

```
+ (NSEvent *)eventWithEventRef:(const void *)eventRef
```

Parameters

eventRef

The EventRef opaque type to be associated with the created NSEvent object.

Return Value
An autoreleased NSEvent object corresponding to *eventRef* or nil if *eventRef* cannot be converted into an equivalent NSEvent object.

Discussion
This method is valid for all events. The created NSEvent object retains the EventRef object and is released when the NSEvent object is freed.

Availability
Available in Mac OS X v10.5 and later.

See Also
- eventRef (page 1079)

Declared In
NSEvent.h

isMouseCoalescingEnabled

Indicates whether mouse-movement event coalescing is enabled.

```
+ (BOOL)isMouseCoalescingEnabled
```

Return Value
YES if mouse-movement event coalescing is enabled, NO if it is disabled.

Availability
Available in Mac OS X v10.5 and later.

See Also
+ setMouseCoalescingEnabled: (page 1069)

Declared In
NSEvent.h

keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode:

Returns a new NSEvent object describing a key event.

```
+ (NSEvent *)keyEventWithType:(NSEventType)type location:(NSPoint)location
    modifierFlags:(NSUInteger)flags timestamp:(NSTimeInterval)time
    windowNumber:(NSInteger)windowNum context:(NSGraphicsContext *)context
    characters:(NSString *)characters charactersIgnoringModifiers:(NSString
    *)unmodCharacters isARepeat:(BOOL)repeatKey keyCode:(unsigned short)code
```

Parameters

type

One of the following event-type constants: `NSKeyDown`, `NSKeyUp`, `NSFlagsChanged`. If anything else is specified, an `NSInternalInconsistencyException` is raised.

location

The cursor location in the base coordinate system of the window specified by *windowNum*.

flags

An integer bit field containing any of the modifier key masks described in "Constants" (page 1091), combined using the C bitwise OR operator.

time

The time the event occurred in seconds since system startup.

windowNum

An integer that identifies the window device associated with the event, which is associated with the `NSWindow` that will receive the event.

context

The display graphics context of the event.

characters

A string of characters associated with the key event. Though most key events contain only one character, it is possible for a single keypress to generate a series of characters.

unmodCharacters

The string of characters generated by the key event as if no modifier key had been pressed (except for Shift). This argument is useful for getting the "basic" key value in a hardware-independent manner.

repeatKey

`YES` if the key event is a repeat caused by the user holding the key down, `NO` if the key event is new.

code

A number that identifies the keyboard key associated with the key event. Its value is hardware-independent.

Return Value

The created `NSEvent` instance or `nil` if the instance could not be created.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `characters` (page 1074)
- `charactersIgnoringModifiers` (page 1074)
- `isARepeat` (page 1079)
- `keyCode` (page 1080)

Declared In

`NSEvent.h`

keyRepeatDelay

Returns the length of time a key must be held down in order to generate the first key repeat event.

```
+ (NSTimeInterval)keyRepeatDelay
```

Return Value
The delay interval, in seconds.

Discussion
This is a system setting, overriding this method will have no effect.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSEvent.h`

keyRepeatInterval

Returns the length between subsequent key repeat events being posted.

```
+ (NSTimeInterval)keyRepeatInterval
```

Return Value
The repeat interval, in seconds.

Discussion
This is a system setting, overriding this method will have no effect.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSEvent.h`

modifierFlags

Returns the currently pressed modifier flags.

```
+ (NSUInteger)modifierFlags
```

Return Value
A mask of the current modifiers using the values in "Modifier Flags" (page 1100).

Discussion
This returns the state of devices combined with synthesized events at the moment, independent of which events have been delivered via the event stream.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
ClockControl

GLUT
LightTable
Rulers
Sketch-112

Declared In
NSEvent.h

mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure:

Returns a new NSEvent object describing a mouse-down, -up, -moved, or -dragged event.

```
+ (NSEvent *)mouseEventWithType:(NSEventType)type location:(NSPoint)location
    modifierFlags:(NSUInteger)flags timestamp:(NSTimeInterval)time
    windowNumber:(NSInteger)windowNum context:(NSGraphicsContext *)context
    eventNumber:(NSInteger)eventNumber clickCount:(NSInteger)clickNumber
    pressure:(float)pressure
```

Parameters

type
One of the modifier key masks described in "Constants" (page 1091), or an NSInternalInconsistencyException is raised.

location
The cursor location in the base coordinate system of the window specified by *windowNum*.

flags
An integer bit field containing any of the modifier key masks described in "Constants" (page 1091), combined using the C bitwise OR operator.

time
The time the event occurred in seconds since system startup.

windowNum
An integer that identifies the window device associated with the event, which is associated with the NSWindow that will receive the event.

context
The display graphics context of the event.

eventNumber
An identifier for the new event. It's normally taken from a counter for mouse events, which continually increases as the application runs.

clickNumber
The number of mouse clicks associated with the mouse event.

pressure
A value from 0.0 to 1.0 indicating the pressure applied to the input device on a mouse event, used for an appropriate device such as a graphics tablet. For devices that aren't pressure-sensitive, the value should be either 0.0 or 1.0.

Return Value
The created NSEvent instance or nil if the instance could not be created.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `clickCount` (page 1075)
- `eventNumber` (page 1078)
- `pressure` (page 1083)

Declared In
`NSEvent.h`

mouseLocation

Reports the current mouse position in screen coordinates.

```
+ (NSPoint)mouseLocation
```

Return Value
The current mouse location in screen coordinates.

Discussion
This method is similar to the `NSWindow` method `mouseLocationOutsideOfEventStream` (page 3348). It returns the location regardless of the current event or pending events. The difference between these methods is that `mouseLocationOutsideOfEventStream` returns a point in the receiving window's coordinates and `mouseLocation` returns the same information in screen coordinates.

> **Note:** The `y` coordinate of the returned point will never be less than `1`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ImageMap
ImageMapExample
PhotoSearch
Quartz Composer Matrix
UIElementInspector

Declared In
`NSEvent.h`

otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2:

Returns a new NSEvent object describing a custom event.

```
+ (NSEvent *)otherEventWithType:(NSEventType)type location:(NSPoint)location
    modifierFlags:(NSUInteger)flags timestamp:(NSTimeInterval)time
    windowNumber:(NSInteger)windowNum context:(NSGraphicsContext *)context
    subtype:(short)subtype data1:(NSInteger)data1 data2:(NSInteger)data2
```

Parameters

type

One of the following event-type constants:

`NSAppKitDefined`

`NSSystemDefined`

`NSApplicationDefined`

`NSPeriodic`

If *type* is anything else, an `NSInternalInconsistencyException` is raised. Your code should only create events of type `NSApplicationDefined`.

location

The cursor location in the base coordinate system of the window specified by *windowNum*.

flags

An integer bit field containing any of the modifier key masks described in "Constants" (page 1091), combined using the C bitwise OR operator.

time

The time the event occurred in seconds since system startup.

windowNum

An integer that identifies the window device associated with the event, which is associated with the `NSWindow` that will receive the event.

context

The display graphics context of the event.

subtype

A numeric identifier that further differentiates custom events of types `NSAppKitDefined`, `NSSystemDefined`, and `NSApplicationDefined`. `NSPeriodic` events don't use this attribute.

data1

Additional data associated with the event. `NSPeriodic` events don't use these attributes.

data2

Additional data associated with the event. `NSPeriodic` events don't use these attributes.

Return Value

The created `NSEvent` object or `nil` if the object couldn't be created.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `subtype` (page 1084)
- `data1` (page 1076)
- `data2` (page 1076)

Declared In

`NSEvent.h`

pressedMouseButtons

Returns the indices of the currently depressed mouse buttons.

```
+ (NSUInteger)pressedMouseButtons
```

Return Value
The indices of the currently depressed mouse buttons.

Discussion
A return value of `1 << 0` corresponds to left the mouse, `1 << 1` corresponds to the right mouse, `1<< n, n >=2` to other mouse buttons.

This returns the state of devices combined with synthesized events at the moment, independent of which events have been delivered via the event stream, so this method is not suitable for tracking.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSEvent.h`

removeMonitor:

Remove the specified event monitor.

```
+ (void)removeMonitor:(id)eventMonitor
```

Parameters

eventMonitor
The event handler object to remove.

Discussion
You must ensure that *eventMonitor* is removed only once. Removing the same *eventMonitor* instance multiple times results in an over-release condition, even in a Garbage Collected environment

Availability
Available in Mac OS X v10.6 and later.

See Also
`+ addGlobalMonitorForEventsMatchingMask:handler:` (page 1058)
`+ addLocalMonitorForEventsMatchingMask:handler:` (page 1060)
`+ removeMonitor:` (page 1069)

Related Sample Code
AnimatedTableView

Declared In
`NSEvent.h`

setMouseCoalescingEnabled:

Controls whether mouse-movement event coalescing is enabled.

```
+ (void)setMouseCoalescingEnabled:(BOOL)flag
```

Parameters

flag
`YES` to enable mouse-movement event coalescing, `NO` to disable it.

Discussion
This method affects mouse-moved, mouse-dragged, and tablet events. Mouse-movement event coalescing is enabled by default.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ isMouseCoalescingEnabled` (page 1063)

Declared In
`NSEvent.h`

startPeriodicEventsAfterDelay:withPeriod:

Begins generating periodic events for the current thread.

```
+ (void)startPeriodicEventsAfterDelay:(NSTimeInterval)delaySeconds
    withPeriod:(NSTimeInterval)periodSeconds
```

Parameters

delaySeconds
The number of seconds that `NSEvent` should wait before beginning to generate periodic events.

periodSeconds
The period in seconds between the generated events.

Discussion
Raises an `NSInternalInconsistencyException` if periodic events are already being generated for the current thread. This method is typically used in a modal loop while tracking mouse-dragged events.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ stopPeriodicEvents` (page 1070)

Related Sample Code
Rulers

Declared In
`NSEvent.h`

stopPeriodicEvents

Stops generating periodic events for the current thread and discards any periodic events remaining in the queue.

```
+ (void)stopPeriodicEvents
```

Discussion
This message is ignored if periodic events aren't currently being generated.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ startPeriodicEventsAfterDelay:withPeriod:` (page 1070)

Related Sample Code
Rulers

Declared In
`NSEvent.h`

Instance Methods

absoluteX

Reports the absolute x coordinate of a pointing device on its tablet at full tablet resolution.

```
- (NSInteger)absoluteX
```

Discussion
For the coordinate to be valid, the receiver should represent an event generated by a tablet pointing device (otherwise 0 is returned). This method is valid only for mouse events with a subtype of `NSTabletPointEventSubtype` and for events of type `NSTabletPoint`. Use this value if you want to scale from tablet location to screen location yourself; otherwise use the class method `mouseLocation` (page 1067) or the instance method `locationInWindow` (page 1080).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- absoluteY` (page 1071)
`- absoluteZ` (page 1072)

Declared In
`NSEvent.h`

absoluteY

Reports the absolute y coordinate of a pointing device on its tablet at full tablet resolution.

```
- (NSInteger)absoluteY
```

Discussion
For the coordinate to be valid, the receiver should represent an event generated by a tablet pointing device (otherwise 0 is returned). This method is valid only for mouse events with a subtype of `NSTabletPointEventSubtype` and for events of type `NSTabletPoint`. Use this value if you want to scale from tablet location to screen location yourself; otherwise use the class method `mouseLocation` (page 1067) or the instance method `locationInWindow` (page 1080).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `absoluteX` (page 1071)
- `absoluteZ` (page 1072)

Declared In
`NSEvent.h`

absoluteZ

Reports the absolute z coordinate of pointing device on its tablet at full tablet resolution.

```
- (NSInteger)absoluteZ
```

Discussion
For the coordinate to be valid, the receiver should represent an event generated by a tablet pointing device (otherwise 0 is returned). The z coordinate does not represent pressure. It registers the depth coordinate returned by some tablet devices with wheels; if the device is something other than these, 0 is returned. This method is valid only for mouse events with a subtype of `NSTabletPointEventSubtype` and for events of type `NSTabletPoint`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `absoluteX` (page 1071)
- `absoluteY` (page 1071)

Declared In
`NSEvent.h`

buttonMask

Returns a bit mask identifying the buttons pressed when the tablet event represented by the receiver was generated.

```
- (NSUInteger)buttonMask
```

Discussion
Use one or more of the button-mask constants described in "Constants" (page 1091) to determine which buttons of the pointing device are pressed. This method is valid only for mouse events with a subtype of `NSTabletPointEventSubtype` and for events of type `NSTabletPoint`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSEvent.h`

buttonNumber

Returns the button number for the mouse button that generated an `NSOtherMouse...` event.

```
- (NSInteger)buttonNumber
```

Discussion
This method is intended for use with the `NSOtherMouseDown`, `NSOtherMouseUp`, and `NSOtherMouseDragged` events, but will return values for `NSLeftMouse...` and `NSRightMouse...` events also.

Availability
Available in Mac OS X v10.1 and later.

Declared In
`NSEvent.h`

capabilityMask

Returns a mask whose set bits indicate the capabilities of the tablet device that generated the event represented by the receiver.

```
- (NSUInteger)capabilityMask
```

Discussion
These bits are vendor-defined. This method is valid only for mouse events with a subtype of `NSTabletProximityEventSubtype` and for events of type `NSTabletProximity`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSEvent.h`

CGEvent

Returns a Core Graphics event object corresponding to the receiver.

```
- (CGEventRef)CGEvent
```

Discussion
The returned `CGEventRef` opaque type is autoreleased. If no `CGEventRef` object corresponding to the `NSEvent` object can be created, this method returns `NULL`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ eventWithCGEvent:` (page 1062)

Declared In
`NSEvent.h`

characters

Returns the characters associated with the receiving key-up or key-down event.

```
- (NSString *)characters
```

Discussion
These characters are derived from a keyboard mapping that associates various key combinations with Unicode characters. Raises an `NSInternalInconsistencyException` if sent to any other kind of event object.

This method returns an empty string for dead keys, such as Option-e. However, for a key combination such as Option-Shift-e this method returns the standard accent ("´").

For a list of constants corresponding to commonly-used Unicode characters, see *NSText Class Reference*.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- charactersIgnoringModifiers` (page 1074)
`+ keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode:` (page 1063)

Related Sample Code
Cocoa OpenGL
CocoaDVDPlayer
FunHouse
GLUT
TrackBall

Declared In
`NSEvent.h`

charactersIgnoringModifiers

Returns the characters generated by the receiving key event as if no modifier key (except for Shift) applies.

```
- (NSString *)charactersIgnoringModifiers
```

Discussion
Raises an `NSInternalInconsistencyException` if sent to a nonkey event.

This method returns the non-modifier key character pressed for dead keys, such as Option-e. For example, Option-e (no shift key) returns an “e" for this method, whereas the `characters` (page 1074) method returns an empty string.

This method is useful for determining “basic” key values in a hardware-independent manner, enabling such features as keyboard equivalents defined in terms of modifier keys plus character keys. For example, to determine if the user typed Alt-S, you don’t have to know whether Alt-S generates a German double ess, an integral sign, or a section symbol. You simply examine the string returned by this method along with the event’s modifier flags, checking for “s” and `NSAlternateKeyMask`.

For a list of constants corresponding to commonly-used Unicode characters, see *NSText Class Reference*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `characters` (page 1074)
- `modifierFlags` (page 1082)
+ keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode: (page 1063)

Related Sample Code
DragItemAround
LayerBackedOpenGLView
NSOpenGL Fullscreen
OpenCL NBody Simulation Example
PDFKitLinker2

Declared In
`NSEvent.h`

clickCount

Returns the number of mouse clicks associated with the receiver, which represents a mouse-down or mouse-up event.

```
- (NSInteger)clickCount
```

Discussion
Raises an `NSInternalInconsistencyException` if sent to a nonmouse event.

Returns 0 for a mouse-up event if a time threshold has passed since the corresponding mouse-down event. This is because if this time threshold passes before the mouse button is released, it is no longer considered a mouse click, but a mouse-down event followed by a mouse-up event.

The return value of this method is meaningless for events other than mouse-down or mouse-up events.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure: (page 1066)

Declared In
`NSEvent.h`

context

Returns the display graphics context of the receiver.

```
- (NSGraphicsContext *)context
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSEvent.h

data1

Returns additional data associated with the receiver.

```
- (NSInteger)data1
```

Discussion
The value returned by this method is dependent on the event type, and is defined by the originator of the event. Raises an `NSInternalInconsistencyException` if sent to an event not of type `NSAppKitDefined`, `NSSystemDefined`, `NSApplicationDefined`, or `NSPeriodic`.

`NSPeriodic` events don't use this attribute.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `data2` (page 1076)
- `subtype` (page 1084)
+ `otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2:` (page 1067)

Declared In
NSEvent.h

data2

Returns additional data associated with the receiver.

```
- (NSInteger)data2
```

Discussion
The value returned by this method is dependent on the event type, and is defined by the originator of the event. Raises an `NSInternalInconsistencyException` if sent to an event not of type `NSAppKitDefined`, `NSSystemDefined`, `NSApplicationDefined`, or `NSPeriodic`.

`NSPeriodic` events don't use this attribute.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `data1` (page 1076)
- `subtype` (page 1084)
+ `otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2:` (page 1067)

Declared In
NSEvent.h

deltaX

Returns the x-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.

```
- (CGFloat)deltaX
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- deltaY (page 1077)
- deltaZ (page 1077)

Declared In
NSEvent.h

deltaY

Returns the y-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.

```
- (CGFloat)deltaY
```

Discussion
The behavior of this method may seem counter-intuitive: as the mouse moves up the screen, the value is negative; and as it moves down the screen, the value is positive. The reason for this behavior is that NSEvent computes this delta value in device space, which is flipped, but both the screen and the window's base coordinate system are not flipped.

Availability
Available in Mac OS X v10.0 and later.

See Also
- deltaX (page 1077)
- deltaZ (page 1077)

Declared In
NSEvent.h

deltaZ

Returns the z-coordinate change for a scroll wheel, mouse-move, or mouse-drag event.

```
- (CGFloat)deltaZ
```

Discussion
This value is typically 0.0.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `deltaX` (page 1077)
- `deltaY` (page 1077)

Related Sample Code
Cocoa OpenGL
TrackBall

Declared In
`NSEvent.h`

deviceID

Returns a special identifier that is used to match tablet-pointer events with the tablet-proximity event represented by the receiver.

```
- (NSUInteger)deviceID
```

Discussion
All tablet-pointer events generated in the period between the device entering and leaving tablet proximity have the same device ID. This message is valid only for mouse events with subtype `NSTabletPointEventSubtype` or `NSTabletProximityEventSubtype`, and for `NSTabletPoint` and `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pointingDeviceID` (page 1082)
- `systemTabletID` (page 1085)
- `tabletID` (page 1085)

Declared In
`NSEvent.h`

eventNumber

Returns the counter value of the latest mouse or tracking-rectangle event object; every system-generated mouse and tracking-rectangle event increments this counter.

```
- (NSInteger)eventNumber
```

Discussion
Raises an `NSInternalInconsistencyException` if sent to any other type of event object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:` (page 1061)

+ mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure: (page 1066)

Declared In
NSEvent.h

eventRef

Returns the Carbon type associated with the receiver for representing an event.

```
- (const void *)eventRef
```

Return Value
Returns an EventRef opaque type corresponding to the receiver. User-input events typically are created with an associated EventRef. An NSEvent object created through other means creates an EventRef in this method if that is necessary and possible. If there is no equivalent NSEvent for the receiver, this method returns NULL.

Discussion
This method is valid for all types of events. The EventRef object is retained by the receiver, so it is valid as long as the NSEvent object is valid, and is released when the NSEvent object is freed. You can use RetainEvent to extend the lifetime of the EventRef object, with a corresponding ReleaseEvent when you are done with it.

Availability
Available in Mac OS X v10.5 and later.

See Also
+ eventWithEventRef: (page 1063)

Declared In
NSEvent.h

isARepeat

Returns YES if the receiving key event is a repeat caused by the user holding the key down, NO if the key event is new.

```
- (BOOL)isARepeat
```

Discussion
Raises an NSInternalInconsistencyException if sent to an NSFlagsChanged event or other nonkey event.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode: (page 1063)

Declared In
NSEvent.h

isEnteringProximity

Returns `YES` to indicate that a pointing device is entering the proximity of its tablet and `NO` when it is leaving it.

```
- (BOOL)isEnteringProximity
```

Discussion
This method is valid for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSEvent.h`

keyCode

Returns the virtual key code for the keyboard key associated with the receiving key event.

```
- (unsigned short)keyCode
```

Return Value
The virtual key code. The returned value is hardware-independent. The value returned is the same as the value returned in the `kEventParamKeyCode` when using Carbon Events.

Discussion
Raises an `NSInternalInconsistencyException` if sent to a non-key event.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode: (page 1063)

Related Sample Code
JSPong
NURBSSurfaceVertexProg
SurfaceVertexProgram
Vertex Optimization

Declared In
`NSEvent.h`

locationInWindow

Returns the receiver's location in the base coordinate system of the associated window.

```
- (NSPoint)locationInWindow
```

Discussion
For non-mouse events the return value of this method is undefined.

With `NSMouseMoved` (page 1094) and possibly other events, the receiver can have a `nil` window (that is, `window` (page 1090) returns nil). In this case, `locationInWindow` returns the event location in screen coordinates.

In a method of a custom view that handles mouse events, you commonly use the `locationInWindow` method in conjunction with the NSView method `convertPoint:fromView:` (page 3155)to get the mouse location in the view's coordinate system. For example:

```
NSPoint event_location = [theEvent locationInWindow];
NSPoint local_point = [self convertPoint:event_location fromView:nil];
```

Note: The y coordinate in the returned point starts from a base of 1, not 0.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- window` (page 1090)

Related Sample Code
Cocoa OpenGL
GLSL Showpiece Lite
LightTable
Sketch+Accessibility
Sketch-112

Declared In
`NSEvent.h`

magnification

Returns the change in magnification.

```
- (CGFloat)magnification
```

Return Value
The change in magnification that should be added to the current scaling of an item to achieve the new scale factor.

Discussion
This message is valid for events of type `NSEventTypeMagnify` (page 1096).

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSEvent.h`

modifierFlags

Returns an integer bit field indicating the modifier keys in effect for the receiver.

```
- (NSUInteger)modifierFlags
```

Discussion
You can examine individual flag settings using the C bitwise AND operator with the predefined key masks described in "Constants" (page 1091). The lower 16 bits of the modifier flags are reserved for device-dependent bits.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSEvent.h`

pointingDeviceID

Returns the index of the pointing device currently in proximity with the tablet.

```
- (NSUInteger)pointingDeviceID
```

Discussion
This index is significant for multimode (or Dual Tracking) tablets that support multiple concurrent pointing devices; the index is incremented for each pointing device that comes into proximity. Otherwise, zero is always returned. The receiver of this message should be a mouse event object with subtype `NSTabletProximityEventSubtype` or an event of type `NSTabletProximity`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pointingDeviceSerialNumber` (page 1082)
- `pointingDeviceType` (page 1083)
- `systemTabletID` (page 1085)

Declared In
`NSEvent.h`

pointingDeviceSerialNumber

Returns the vendor-assigned serial number of a pointing device of a certain type.

```
- (NSUInteger)pointingDeviceSerialNumber
```

Discussion
Devices of different types, such as a puck and a pen, may have the same serial number. The receiver of this message should be a mouse event object with subtype `NSTabletProximityEventSubtype` or an event of type `NSTabletProximity`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pointingDeviceID` (page 1082)
- `pointingDeviceType` (page 1083)

Declared In
`NSEvent.h`

pointingDeviceType

Returns a `NSPointingDeviceType` constant indicating the kind of pointing device associated with the receiver.

```
- (NSPointingDeviceType)pointingDeviceType
```

Discussion
For example, the device could be a pen, eraser, or cursor pointing device. This method is valid for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events. See "Constants" (page 1091) for descriptions of valid `NSPointingDeviceType` constants.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pointingDeviceSerialNumber` (page 1082)
- `pointingDeviceType` (page 1083)

Declared In
`NSEvent.h`

pressure

Returns a value from 0.0 through 1.0 indicating the pressure applied to the input device (used for appropriate devices).

```
- (float)pressure
```

Discussion
For devices that aren't pressure-sensitive, the value is either 0.0 or 1.0. Raises an `NSInternalInconsistencyException` if sent to a nonmouse event.

For tablet pointing devices that are in proximity, the pressure value is 0.0 if they are not actually touching the tablet. As the device is pressed into the tablet, the value is increased.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ `mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure:` (page 1066)
- `rotation` (page 1084)

Declared In
NSEvent.h

rotation

Returns the rotation in degrees of the tablet pointing device associated with the receiver.

```
- (float)rotation
```

Discussion
Many devices do not support rotation, in which case the returned value is 0.0. This method is valid only for mouse events with subtype `NSTabletPointEventSubtype` and for `NSTabletPoint` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pressure` (page 1083)
- `tilt` (page 1086)

Declared In
NSEvent.h

subtype

Returns the subtype of the receiving event object.

```
- (short)subtype
```

Discussion
Raises an `NSInternalInconsistencyException` if sent to an event not of type `NSAppKitDefined`, `NSSystemDefined`, `NSApplicationDefined`, or `NSPeriodic`.

`NSPeriodic` events don't use this attribute.

This method is also valid for mouse events on Mac OS X v10.4 and later. See "Constants" (page 1091) for the predefined mouse and tablet subtypes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `data1` (page 1076)
- `data2` (page 1076)
- `+ otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2:` (page 1067)

Declared In
NSEvent.h

systemTabletID

Returns the index of the tablet device connected to the system.

```
- (NSUInteger)systemTabletID
```

Discussion
If multiple tablets are connected to the system, the system-tablet ID is incremented for each subsequent one. If there is only one tablet device, its system-tablet ID is zero. The receiver of this message should be a mouse event object with subtype `NSTabletProximityEventSubtype` or an event of type `NSTabletProximity`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pointingDeviceID` (page 1082)
- `tabletID` (page 1085)

Declared In
`NSEvent.h`

tabletID

Returns the USB model identifier of the tablet device associated with the receiver.

```
- (NSUInteger)tabletID
```

Discussion
This method is valid for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pointingDeviceID` (page 1082)
- `systemTabletID` (page 1085)

Declared In
`NSEvent.h`

tangentialPressure

Reports the tangential pressure on the device that generated the event represented by the receiver.

```
- (float)tangentialPressure
```

Discussion
The value returned can range from -1.0 to 1.0. Tangential pressure is also known as barrel pressure. Only some pointing devices support tangential pressure. This method is valid for mouse events with subtype `NSTabletPointEventSubtype` and for `NSTabletPoint` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pressure` (page 1083)

Declared In
`NSEvent.h`

tilt

Reports the scaled tilt values of the pointing device that generated the event represented by the receiver.

```
- (NSPoint)tilt
```

Discussion
The value returned can range from -1.0 to 1.0 for both axes. An x-coordinate value that is negative indicates a tilt to the left and a positive value indicates a tilt to the right; a y-coordinate value that is negative indicates a tilt to the top and a positive value indicates a tilt to the bottom. If the device is perfectly perpendicular to the table surface, the values are 0.0 for both axes. This method is valid for mouse events with subtype `NSTabletPointEventSubtype` and for `NSTabletPoint` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `pressure` (page 1083)
- `rotation` (page 1084)

Declared In
`NSEvent.h`

timestamp

Returns the time the receiver occurred in seconds since system startup.

```
- (NSTimeInterval)timestamp
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSEvent.h`

touchesMatchingPhase:inView:

Returns all the `NSTouch` objects associated with a specific phase.

```
- (NSSet *)touchesMatchingPhase:(NSTouchPhase)phase inView:(NSView *)view
```

Parameters

phase

The touch phase for which you want touches. See "Touch Phases" (page 1091) for the possible values.

view

The view for which touches are wanted. Touches that target this view, or any of the view's descendants will be returned. Passing `nil` as the view gets all touches regardless of their targeted view.

Return Value
A set of applicable `NSTouch` objects.

Discussion
This method is only valid for gesture events (gesture, magnify, swipe, rotate, etc.). Using this method a view can get all of the touches associated with a gesture without overriding the individual touch responder methods.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
LightTable

Declared In
`NSEvent.h`

trackingArea

Returns the `NSTrackingArea` object that generated the event represented by the receiver.

```
- (NSTrackingArea *)trackingArea
```

Return Value
Returns the `NSTrackingArea` object that generated the event represented by the receiver. If the receiver is not a mouse-tracking event (that is, an event of type `NSMouseEntered` (page 1094), `NSMouseExited` (page 1094), or `NSCursorUpdate` (page 1094)), this method raises an `NSInternalInconsistencyException`. This method returns `nil` if the event was generated by a tracking rectangle (pre-Mac OS X version 10.5) instead of a `NSTrackingArea` object.

Discussion
If no `NSTrackingArea` object is associated with the event because the event corresponds to a tracking rectangle installed with the `NSView` method `addTrackingRect:owner:userData:assumeInside:` (page 3142), this method returns `nil`. Note that the `trackingNumber` (page 1087) method returns either an `NSTrackingArea` object or the `NSTrackingRectTag` (page 3251) constant depending on how the event was generated.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSEvent.h`

trackingNumber

Returns the identifier of a mouse-tracking event.

```
- (NSInteger)trackingNumber
```

Discussion
This method returns either an `NSTrackingArea` object or a `NSTrackingRectTag` (page 3251) constant depending on whether the event was generated from an `NSTrackingArea` object or a call to `addTrackingRect:owner:userData:assumeInside:` (page 3142). Valid mouse-tracking methods are of types `NSMouseEntered` (page 1094), `NSMouseExited` (page 1094), and `NSCursorUpdate` (page 1094). This method raises an `NSInternalInconsistencyException` if sent to any other type of event.

The `NSTrackingArea` class is new with Mac OS X version 10.5

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:` (page 1061)
`- trackingArea` (page 1087)

Declared In
`NSEvent.h`

type

Returns the type of the receiving event.

```
- (NSEventType)type
```

Return Value
Returns the event type. The possible values are described in "Event Types" (page 1092).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
People
VBL

Declared In
`NSEvent.h`

uniqueID

Returns the unique identifier of the pointing device that generated the event represented by the receiver.

```
- (unsigned long long)uniqueID
```

Discussion
Also known as tool ID, this is a unique number recorded in the chip inside every pointing device. The unique ID makes it possible to assign a specific pointing device to a specific tablet. You can also use it to "sign" documents or to restrict access to document layers to a specific pointing device. This method is valid for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `vendorDefined` (page 1089)
- `vendorID` (page 1090)

Declared In
`NSEvent.h`

userData

Returns data associated with a mouse-tracking event,

```
- (void *)userData
```

Discussion
The returned data was assigned to the mouse-tracking event when it was set up using the `NSView` method `addTrackingRect:owner:userData:assumeInside:` (page 3142). It is only valid to send this message if the receiver represents an `NSMouseEntered` (page 1094) or `NSMouseExited` (page 1094) event. Raises an `NSInternalInconsistencyException` if sent to any other type of event object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:` (page 1061)

Related Sample Code
MenuItemView
PhotoSearch
TrackIt

Declared In
`NSEvent.h`

vendorDefined

Returns an array of three vendor-defined `NSNumber` objects associated with the pointing-type event represented by the receiver.

```
- (id)vendorDefined
```

Discussion
The `NSNumber` objects encapsulate `short` values that vendors may return for various reasons; see the vendor documentation for details.This method is valid for mouse events with subtype `NSTabletPointEventSubtype` and for `NSTabletPoint` events.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSEvent.h

vendorID

Returns the vendor identifier of the tablet associated with the receiver.

```
- (NSUInteger)vendorID
```

Discussion
The tablet is typically a USB device. This method is valid only for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `tabletID` (page 1085)
- `vendorPointingDeviceType` (page 1090)

Declared In
NSEvent.h

vendorPointingDeviceType

Returns a coded bit field whose set bits indicate the type of pointing device (within a vendor selection) associated with the receiver.

```
- (NSUInteger)vendorPointingDeviceType
```

Discussion
See the vendor documentation for an interpretation of significant bits. This method is valid only for mouse events with subtype `NSTabletProximityEventSubtype` and for `NSTabletProximity` events.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `vendorID` (page 1090)

Declared In
NSEvent.h

window

Returns the window object associated with the receiver.

```
- (NSWindow *)window
```

Discussion
A periodic event, however, has no window; in this case the return value is undefined.

Availability
Available in Mac OS X v10.0 and later.

See Also
- windowNumber (page 1091)

Declared In
NSEvent.h

windowNumber

Returns the identifier for the window device associated with the receiver.

```
- (NSInteger)windowNumber
```

Discussion
A periodic event, however, has no window; in this case the return value is undefined.

Availability
Available in Mac OS X v10.0 and later.

See Also
- window (page 1090)

Declared In
NSEvent.h

Constants

Touch Phases

These constants represent the possible phases during a touch gesture.

```
enum {
    NSTouchPhaseBegan           = 1 << 0,
    NSTouchPhaseMoved           = 1 << 1,
    NSTouchPhaseStationary      = 1 << 2,
    NSTouchPhaseEnded           = 1 << 3,
    NSTouchPhaseCancelled       = 1 << 4,
    NSTouchPhaseTouching = NSTouchPhaseBegan | NSTouchPhaseMoved |
NSTouchPhaseStationary,
    NSTouchPhaseAny             = NSUIntegerMax
};
typedef NSUInteger NSTouchPhase;
```

Constants
NSTouchPhaseBegan
A finger for a given event touched the screen.
Available in Mac OS X v10.6 and later.
Declared in NSTouch.h.

`NSTouchPhaseMoved`
A finger for a given event moved on the screen.
Available in Mac OS X v10.6 and later.
Declared in `NSTouch.h`.

`NSTouchPhaseStationary`
A finger is touching the surface but hasn't moved since the previous event.
Available in Mac OS X v10.6 and later.
Declared in `NSTouch.h`.

`NSTouchPhaseEnded`
A finger for a given event was lifted from the screen.
Available in Mac OS X v10.6 and later.
Declared in `NSTouch.h`.

`NSTouchPhaseCancelled`
The system cancelled tracking for the touch.
Available in Mac OS X v10.6 and later.
Declared in `NSTouch.h`.

`NSTouchPhaseTouching`
An NSTouchPhaseBegan, or NSTouchPhaseMoved, or NSTouchPhaseStationary phase is in progress.
Available in Mac OS X v10.6 and later.
Declared in `NSTouch.h`.

`NSTouchPhaseAny`
Any touch phase.
Available in Mac OS X v10.6 and later.
Declared in `NSTouch.h`.

Event Types

These constants represent various kinds of events. They are returned by `type` (page 1088) and are used as the first argument to the methods `enterExitEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:trackingNumber:userData:` (page 1061), `keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode:` (page 1063), `mouseEventWithType:location:modifierFlags:timestamp:windowNumber:context:eventNumber:clickCount:pressure:` (page 1066), and `otherEventWithType:location:modifierFlags:timestamp:windowNumber:context:subtype:data1:data2:` (page 1067).

```
enum {
    NSLeftMouseDown      = 1,
    NSLeftMouseUp        = 2,
    NSRightMouseDown     = 3,
    NSRightMouseUp       = 4,
    NSMouseMoved         = 5,
    NSLeftMouseDragged   = 6,
    NSRightMouseDragged  = 7,
    NSMouseEntered       = 8,
    NSMouseExited        = 9,
    NSKeyDown            = 10,
    NSKeyUp              = 11,
    NSFlagsChanged       = 12,
    NSAppKitDefined      = 13,
    NSSystemDefined      = 14,
    NSApplicationDefined = 15,
    NSPeriodic           = 16,
    NSCursorUpdate       = 17,
    NSScrollWheel        = 22,
    NSTabletPoint        = 23,
    NSTabletProximity    = 24,
    NSOtherMouseDown     = 25,
    NSOtherMouseUp       = 26,
    NSOtherMouseDragged  = 27
    NSEventTypeGesture   = 29,
    NSEventTypeMagnify   = 30,
    NSEventTypeSwipe     = 31,
    NSEventTypeRotate    = 18,
    NSEventTypeBeginGesture = 19,
    NSEventTypeEndGesture   = 20
};
typedef NSUInteger NSEventType;
```

Constants

`NSLeftMouseDown`

See “Handling Mouse Events” in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSLeftMouseUp`

See “Handling Mouse Events” in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightMouseDown`

See “Handling Mouse Events” in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightMouseUp`

See “Handling Mouse Events” in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSOtherMouseDown`

See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.1 and later.

Declared in `NSEvent.h`.

`NSOtherMouseUp`

See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.1 and later.

Declared in `NSEvent.h`.

`NSMouseMoved`

See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSLeftMouseDragged`

See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightMouseDragged`

See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSOtherMouseDragged`

See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.1 and later.

Declared in `NSEvent.h`.

`NSMouseEntered`

See "Mouse-Tracking and Cursor-Update Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSMouseExited`

See "Mouse-Tracking and Cursor-Update Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSCursorUpdate`

See "Mouse-Tracking and Cursor-Update Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSKeyDown`

See "Handling Key Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSKeyUp`
See "Handling Key Events" in *Cocoa Event-Handling Guide*.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSFlagsChanged`
See "Handling Key Events" in *Cocoa Event-Handling Guide*.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSAppKitDefined`
See "Event Objects and Types" in *Cocoa Event-Handling Guide*.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSSystemDefined`
See "Event Objects and Types" in *Cocoa Event-Handling Guide*.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSApplicationDefined`
See "Event Objects and Types" in *Cocoa Event-Handling Guide*.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSPeriodic`
See "Event Objects and Types" in *Cocoa Event-Handling Guide*.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSScrollWheel`
See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSTabletPoint`
An event representing the current state of a tablet pointing device, including its location, pressure, and tilt.
Available in Mac OS X v10.4 and later.
Declared in `NSEvent.h`.

`NSTabletProximity`
An event representing the proximity of a pointing device to its tablet.
Available in Mac OS X v10.4 and later.
Declared in `NSEvent.h`.

`NSEventTypeGesture`
An event that represents some type of gesture such as `NSEventTypeMagnify`, `NSEventTypeSwipe`, `NSEventTypeRotate`, `NSEventTypeBeginGesture`, or `NSEventTypeEndGesture`.
Available in Mac OS X v10.6 and later.
Declared in `NSEvent.h`.

`NSEventTypeMagnify`
: An event representing a pinch open or pinch close gesture.

: Available in Mac OS X v10.6 and later.

: Declared in `NSEvent.h`.

`NSEventTypeSwipe`
: An event representing a swipe gesture.

: Available in Mac OS X v10.6 and later.

: Declared in `NSEvent.h`.

`NSEventTypeRotate`
: An event representing a rotation gesture.

: Available in Mac OS X v10.6 and later.

: Declared in `NSEvent.h`.

`NSEventTypeBeginGesture`
: An event that represents a gesture beginning.

: Available in Mac OS X v10.6 and later.

: Declared in `NSEvent.h`.

`NSEventTypeEndGesture`
: An event that represents a gesture ending.

: Available in Mac OS X v10.6 and later.

: Declared in `NSEvent.h`.

Masks for event types

These constants are masks for the events defined in "Event Types" (page 1092). Pass them to the `NSCell` method `sendActionOn:` (page 576) to specify when an `NSCell` should send its action message.

```
enum {
   NSLeftMouseDownMask      = 1 << NSLeftMouseDown,
   NSLeftMouseUpMask        = 1 << NSLeftMouseUp,
   NSRightMouseDownMask     = 1 << NSRightMouseDown,
   NSRightMouseUpMask       = 1 << NSRightMouseUp,
   NSMouseMovedMask         = 1 << NSMouseMoved,
   NSLeftMouseDraggedMask   = 1 << NSLeftMouseDragged,
   NSRightMouseDraggedMask  = 1 << NSRightMouseDragged,
   NSMouseEnteredMask       = 1 << NSMouseEntered,
   NSMouseExitedMask        = 1 << NSMouseExited,
   NSKeyDownMask            = 1 << NSKeyDown,
   NSKeyUpMask              = 1 << NSKeyUp,
   NSFlagsChangedMask       = 1 << NSFlagsChanged,
   NSAppKitDefinedMask      = 1 << NSAppKitDefined,
   NSSystemDefinedMask      = 1 << NSSystemDefined,
   NSApplicationDefinedMask = 1 << NSApplicationDefined,
   NSPeriodicMask           = 1 << NSPeriodic,
   NSCursorUpdateMask       = 1 << NSCursorUpdate,
   NSScrollWheelMask        = 1 << NSScrollWheel,
   NSTabletPointMask           = 1 << NSTabletPoint,
   NSTabletProximityMask       = 1 << NSTabletProximity,
   NSOtherMouseDownMask     = 1 << NSOtherMouseDown,
   NSOtherMouseUpMask       = 1 << NSOtherMouseUp,
   NSOtherMouseDraggedMask  = 1 << NSOtherMouseDragged,
   NSEventMaskGesture          = 1 << NSEventTypeGesture,
   NSEventMaskMagnify          = 1 << NSEventTypeMagnify,
   NSEventMaskSwipe            = 1U << NSEventTypeSwipe,
   NSEventMaskRotate           = 1 << NSEventTypeRotate,
   NSEventMaskBeginGesture     = 1 << NSEventTypeBeginGesture,
   NSEventMaskEndGesture       = 1 << NSEventTypeEndGesture,
   NSAnyEventMask           = 0xffffffffU
};
NSUInteger NSEventMaskFromType(NSEventType type) { return (1 << type); };
```

Constants

`NSLeftMouseDownMask`

Corresponds to `NSLeftMouseDown`. See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSLeftMouseUpMask`

Corresponds to `NSLeftMouseUp`. See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightMouseDownMask`

Corresponds to `NSRightMouseDown`. See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightMouseUpMask`

Corresponds to `NSRightMouseUp`. See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSOtherMouseDownMask`
Corresponds to `NSOtherMouseDown`. See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.1 and later.

Declared in `NSEvent.h`.

`NSOtherMouseUpMask`
Corresponds to `NSOtherMouseUp`. See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.1 and later.

Declared in `NSEvent.h`.

`NSMouseMovedMask`
Corresponds to `NSMouseMoved`. See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSLeftMouseDraggedMask`
Corresponds to `NSLeftMouseDragged`. See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightMouseDraggedMask`
Corresponds to `NSRightMouseDragged`. See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSOtherMouseDraggedMask`
Corresponds to `NSOtherMouseDragged`. See "Handling Mouse Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.1 and later.

Declared in `NSEvent.h`.

`NSMouseEnteredMask`
Corresponds to `NSMouseEntered`. See See "Mouse-Tracking and Cursor-Update Events" in *Cocoa Event-Handling Guide*..

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSMouseExitedMask`
Corresponds to `NSMouseExited`. See See "Mouse-Tracking and Cursor-Update Events" in *Cocoa Event-Handling Guide*..

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSCursorUpdateMask`
Corresponds to `NSCursorUpdate`. See See "Mouse-Tracking and Cursor-Update Events" in *Cocoa Event-Handling Guide*..

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSKeyDownMask`
Corresponds to `NSKeyDown`. See "Handling Key Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSKeyUpMask`
Corresponds to `NSKeyUp`. See "Handling Key Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSFlagsChangedMask`
Corresponds to `NSFlagsChanged`. See "Handling Key Events" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSAppKitDefinedMask`
Corresponds to `NSAppKitDefined`. See "Event Objects and Types" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSSystemDefinedMask`
Corresponds to `NSSystemDefined`. See "Event Objects and Types" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSApplicationDefinedMask`
Corresponds to `NSApplicationDefined`. See "Event Objects and Types" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSPeriodicMask`
Corresponds to `NSPeriodic`. See "Event Objects and Types" in *Cocoa Event-Handling Guide*.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSScrollWheelMask`
Corresponds to `NSScrollWheel`. See "Handling Mouse Events" in *Cocoa Event-Handling Guide*..

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSTabletPointMask`
Corresponds to `NSTabletPoint`.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSTabletProximityMask`
Corresponds to `NSTabletProximity`.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSEventMaskGesture`
Corresponds to `NSEventTypeGesture`.

Available in Mac OS X v10.6 and later.

Declared in `NSEvent.h`.

`NSEventMaskMagnify`
Corresponds to `NSEventTypeMagnify`.
Available in Mac OS X v10.6 and later.
Declared in `NSEvent.h`.

`NSEventMaskSwipe`
Corresponds to `NSEventTypeSwipe`.
Available in Mac OS X v10.6 and later.
Declared in `NSEvent.h`.

`NSEventMaskRotate`
Corresponds to `NSEventTypeRotate`.
Available in Mac OS X v10.6 and later.
Declared in `NSEvent.h`.

`NSEventMaskBeginGesture`
Corresponds to `NSEventTypeBeginGesture`.
Available in Mac OS X v10.6 and later.
Declared in `NSEvent.h`.

`NSEventMaskEndGesture`
Corresponds to `NSEventTypeEndGesture`.
Available in Mac OS X v10.6 and later.
Declared in `NSEvent.h`.

`NSAnyEventMask`
Corresponds to any event mask.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

Modifier Flags

The following constants (except for `NSDeviceIndependentModifierFlagsMask`) represent device-independent bits found in event modifier flags:

```
enum {
    NSAlphaShiftKeyMask = 1 << 16,
    NSShiftKeyMask      = 1 << 17,
    NSControlKeyMask    = 1 << 18,
    NSAlternateKeyMask  = 1 << 19,
    NSCommandKeyMask    = 1 << 20,
    NSNumericPadKeyMask = 1 << 21,
    NSHelpKeyMask       = 1 << 22,
    NSFunctionKeyMask   = 1 << 23,
    NSDeviceIndependentModifierFlagsMask = 0xffff0000U
};
```

Constants

`NSAlphaShiftKeyMask`
Set if Caps Lock key is pressed.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSShiftKeyMask`
Set if Shift key is pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSControlKeyMask`
Set if Control key is pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSAlternateKeyMask`
Set if Option or Alternate key is pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSCommandKeyMask`
Set if Command key is pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSNumericPadKeyMask`
Set if any key in the numeric keypad is pressed. The numeric keypad is generally on the right side of the keyboard. This is also set if any of the arrow keys are pressed (`NSUpArrowFunctionKey` (page 1107), `NSDownArrowFunctionKey` (page 1107), `NSLeftArrowFunctionKey` (page 1107), and `NSRightArrowFunctionKey` (page 1107)).

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSHelpKeyMask`
Set if the Help key is pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSFunctionKeyMask`
Set if any function key is pressed. The function keys include the F keys at the top of most keyboards (F1, F2, and so on) and the navigation keys in the center of most keyboards (Help, Forward Delete, Home, End, Page Up, Page Down, and the arrow keys).

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSDeviceIndependentModifierFlagsMask`
Used to retrieve only the device-independent modifier flags, allowing applications to mask off the device-dependent modifier flags, including event coalescing information.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

NSPointingDeviceType

The following constants represent pointing-device types for `NSTabletProximity` events or mouse events with subtype `NSTabletProximityEventSubtype`. The `pointingDeviceType` (page 1083) method returns one of these constants.

```
typedef enum {
    NSUnknownPointingDevice = NX_TABLET_POINTER_UNKNOWN,
    NSPenPointingDevice     = NX_TABLET_POINTER_PEN,
    NSCursorPointingDevice  = NX_TABLET_POINTER_CURSOR,
    NSEraserPointingDevice  = NX_TABLET_POINTER_ERASER
} NSPointingDeviceType;
```

Constants

`NSUnknownPointingDevice`
Represents an unknown type of pointing device.
Available in Mac OS X v10.4 and later.
Declared in `NSEvent.h`.

`NSPenPointingDevice`
Represents the tip end of a stylus-like pointing device.
Available in Mac OS X v10.4 and later.
Declared in `NSEvent.h`.

`NSCursorPointingDevice`
Represents a cursor (or puck-like) pointing device.
Available in Mac OS X v10.4 and later.
Declared in `NSEvent.h`.

`NSEraserPointingDevice`
Represents the eraser end of a stylus-like pointing device.
Available in Mac OS X v10.4 and later.
Declared in `NSEvent.h`.

Mouse-event subtypes

The following constants represent mouse-event subtypes for mouse and tablet events (accessed with the `subtype` (page 1084) method).

```
enum {
    NSMouseEventSubtype           = NX_SUBTYPE_DEFAULT,
    NSTabletPointEventSubtype     = NX_SUBTYPE_TABLET_POINT,
    NSTabletProximityEventSubtype = NX_SUBTYPE_TABLET_PROXIMITY
    NSTouchEventSubtype             = NX_SUBTYPE_MOUSE_TOUCH
};
```

Constants

`NSMouseEventSubtype`
Indicates a purely mouse event.
Available in Mac OS X v10.4 and later.
Declared in `NSEvent.h`.

`NSTabletPointEventSubtype`
Indicates a tablet-pointer event; see description of `NSTabletPoint`.
Available in Mac OS X v10.4 and later.
Declared in `NSEvent.h`.

`NSTabletProximityEventSubtype`
: Indicates a tablet-proximity event; see description of `NSTabletProximity`.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSTouchEventSubtype`
: Indicates a touch event subtype.

Available in Mac OS X v10.6 and later.

Declared in `NSEvent.h`.

Tablet event masks

The following constants represent button masks for `NSTabletPoint` events or mouse events with subtype `NSTabletPointEventSubtype`. The `buttonMask` (page 1072) method returns a bit mask, which you test with one or more of these constants to determine the state of the buttons on a tablet pointing device.

```
enum {
    NSPenTipMask =       NX_TABLET_BUTTON_PENTIPMASK,
    NSPenLowerSideMask = NX_TABLET_BUTTON_PENLOWERSIDEMASK,
    NSPenUpperSideMask = NX_TABLET_BUTTON_PENUPPERSIDEMASK
};
```

Constants

`NSPenTipMask`
: The pen tip is activated.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSPenLowerSideMask`
: The button on the lower side of the device is activated.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

`NSPenUpperSideMask`
: The button on the upper side of the device is activated.

Available in Mac OS X v10.4 and later.

Declared in `NSEvent.h`.

Types Defined by the Application Kit

These constants represent the types of events defined by the Application Kit.

```
enum {
   NSWindowExposedEventType = 0,
   NSApplicationActivatedEventType = 1,
   NSApplicationDeactivatedEventType = 2,
   NSWindowMovedEventType = 4,
   NSScreenChangedEventType = 8,
   NSAWTEventType = 16
};
```

Constants

`NSWindowExposedEventType`
A non-retained NSWindow has been exposed.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSApplicationActivatedEventType`
The application has been activated.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSApplicationDeactivatedEventType`
The application has been deactivated.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSWindowMovedEventType`
An NSWindow has moved.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSScreenChangedEventType`
An NSWindow has changed screens.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSAWTEventType`
An event type used to support Java applications.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

Power-off event

This constant denotes that the user is turning off the computer.

```
enum {
   NSPowerOffEventType = 1
};
```

Constants

`NSPowerOffEventType`

Specifies that the user is turning off the computer.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

Function-Key Unicodes

These constants represent Unicode characters (0xF700–0xF8FF) that are reserved for function keys on the keyboard. Combined in NSStrings, they are the return values of the NSEvent methods `characters` (page 1074) and `charactersIgnoringModifiers` (page 1074) and may be used in some parameters in the NSEvent method `keyEventWithType:location:modifierFlags:timestamp:windowNumber:context:characters:charactersIgnoringModifiers:isARepeat:keyCode:` (page 1063).

```
enum {
   NSUpArrowFunctionKey = 0xF700,
   NSDownArrowFunctionKey = 0xF701,
   NSLeftArrowFunctionKey = 0xF702,
   NSRightArrowFunctionKey = 0xF703,
   NSF1FunctionKey = 0xF704,
   NSF2FunctionKey = 0xF705,
   NSF3FunctionKey = 0xF706,
   NSF4FunctionKey = 0xF707,
   NSF5FunctionKey = 0xF708,
   NSF6FunctionKey = 0xF709,
   NSF7FunctionKey = 0xF70A,
   NSF8FunctionKey = 0xF70B,
   NSF9FunctionKey = 0xF70C,
   NSF10FunctionKey = 0xF70D,
   NSF11FunctionKey = 0xF70E,
   NSF12FunctionKey = 0xF70F,
   NSF13FunctionKey = 0xF710,
   NSF14FunctionKey = 0xF711,
   NSF15FunctionKey = 0xF712,
   NSF16FunctionKey = 0xF713,
   NSF17FunctionKey = 0xF714,
   NSF18FunctionKey = 0xF715,
   NSF19FunctionKey = 0xF716,
   NSF20FunctionKey = 0xF717,
   NSF21FunctionKey = 0xF718,
   NSF22FunctionKey = 0xF719,
   NSF23FunctionKey = 0xF71A,
   NSF24FunctionKey = 0xF71B,
   NSF25FunctionKey = 0xF71C,
   NSF26FunctionKey = 0xF71D,
   NSF27FunctionKey = 0xF71E,
   NSF28FunctionKey = 0xF71F,
   NSF29FunctionKey = 0xF720,
   NSF30FunctionKey = 0xF721,
   NSF31FunctionKey = 0xF722,
   NSF32FunctionKey = 0xF723,
   NSF33FunctionKey = 0xF724,
   NSF34FunctionKey = 0xF725,
   NSF35FunctionKey = 0xF726,
   NSInsertFunctionKey = 0xF727,
   NSDeleteFunctionKey = 0xF728,
   NSHomeFunctionKey = 0xF729,
   NSBeginFunctionKey = 0xF72A,
   NSEndFunctionKey = 0xF72B,
   NSPageUpFunctionKey = 0xF72C,
   NSPageDownFunctionKey = 0xF72D,
   NSPrintScreenFunctionKey = 0xF72E,
   NSScrollLockFunctionKey = 0xF72F,
   NSPauseFunctionKey = 0xF730,
   NSSysReqFunctionKey = 0xF731,
   NSBreakFunctionKey = 0xF732,
   NSResetFunctionKey = 0xF733,
   NSStopFunctionKey = 0xF734,
   NSMenuFunctionKey = 0xF735,
   NSUserFunctionKey = 0xF736,
   NSSystemFunctionKey = 0xF737,
   NSPrintFunctionKey = 0xF738,
```

```
    NSClearLineFunctionKey = 0xF739,
    NSClearDisplayFunctionKey = 0xF73A,
    NSInsertLineFunctionKey = 0xF73B,
    NSDeleteLineFunctionKey = 0xF73C,
    NSInsertCharFunctionKey = 0xF73D,
    NSDeleteCharFunctionKey = 0xF73E,
    NSPrevFunctionKey = 0xF73F,
    NSNextFunctionKey = 0xF740,
    NSSelectFunctionKey = 0xF741,
    NSExecuteFunctionKey = 0xF742,
    NSUndoFunctionKey = 0xF743,
    NSRedoFunctionKey = 0xF744,
    NSFindFunctionKey = 0xF745,
    NSHelpFunctionKey = 0xF746,
    NSModeSwitchFunctionKey = 0xF747
};
```

Constants

`NSUpArrowFunctionKey`

Up Arrow key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSDownArrowFunctionKey`

Down Arrow key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSLeftArrowFunctionKey`

Left Arrow key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSRightArrowFunctionKey`

Right Arrow key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF1FunctionKey`

F1 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF2FunctionKey`

F2 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF3FunctionKey`

F3 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF4FunctionKey`
F4 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF5FunctionKey`
F5 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF6FunctionKey`
F6 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF7FunctionKey`
F7 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF8FunctionKey`
F8 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF9FunctionKey`
F9 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF10FunctionKey`
F10 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF11FunctionKey`
F11 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF12FunctionKey`
F12 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF13FunctionKey`
F13 key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSF14FunctionKey`
F14 key.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF15FunctionKey`
F15 key.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF16FunctionKey`
F16 key.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF17FunctionKey`
F17 key.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF18FunctionKey`
F18 key.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF19FunctionKey`
F19 key.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF20FunctionKey`
F20 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF21FunctionKey`
F21 keyc. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF22FunctionKey`
F22 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF23FunctionKey`
F23 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF24FunctionKey`
F24 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF25FunctionKey`
F25 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF26FunctionKey`
F26 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF27FunctionKey`
F27 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF28FunctionKey`
F28 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF29FunctionKey`
F29 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF30FunctionKey`
F30 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF31FunctionKey`
F31 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF32FunctionKey`
F32 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF33FunctionKey`
F33 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF34FunctionKey`
F34 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSF35FunctionKey`
F35 key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSInsertFunctionKey`
Insert key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSDeleteFunctionKey`
Forward Delete key.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSHomeFunctionKey`
Home key.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSBeginFunctionKey`
Begin key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSEndFunctionKey`
End key.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSPageUpFunctionKey`
Page Up key.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSPageDownFunctionKey`
Page Down key.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSPrintScreenFunctionKey`
Print Screen key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSScrollLockFunctionKey`
Scroll Lock key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSPauseFunctionKey`
Pause key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSSysReqFunctionKey`
System Request key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSBreakFunctionKey`
Break key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSResetFunctionKey`
Reset key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSStopFunctionKey`
Stop key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSMenuFunctionKey`
Menu key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSUserFunctionKey`
User key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSSystemFunctionKey`
System key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSPrintFunctionKey`
Print key. Not on most Macintosh keyboards.
Available in Mac OS X v10.0 and later.
Declared in `NSEvent.h`.

`NSClearLineFunctionKey`
Clear/Num Lock key.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSClearDisplayFunctionKey`
Clear Display key. Not on most Macintosh keyboards.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSInsertLineFunctionKey`
Insert Line key. Not on most Macintosh keyboards.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSDeleteLineFunctionKey`
Delete Line key. Not on most Macintosh keyboards.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSInsertCharFunctionKey`
Insert Character key. Not on most Macintosh keyboards.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSDeleteCharFunctionKey`
Delete Character key. Not on most Macintosh keyboards.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSPrevFunctionKey`
Previous key. Not on most Macintosh keyboards.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSNextFunctionKey`
Next key. Not on most Macintosh keyboards.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSSelectFunctionKey`
Select key. Not on most Macintosh keyboards.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSExecuteFunctionKey`
Execute key. Not on most Macintosh keyboards.

Available in Mac OS X v10.0 and later.

Declared in `NSEvent.h`.

`NSUndoFunctionKey`
: Undo key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSRedoFunctionKey`
: Redo key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSFindFunctionKey`
: Find key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSHelpFunctionKey`
: Help key.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

`NSModeSwitchFunctionKey`
: Mode Switch key. Not on most Macintosh keyboards.

  Available in Mac OS X v10.0 and later.

  Declared in `NSEvent.h`.

Discussion
Note that some function keys are handled at a lower level and are never seen by your application. They include the Volume Up key, Volume Down key, Volume Mute key, Eject key, and Function key found on many iBook and PowerBook computers.

NSFileWrapper Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	Foundation/NSFileWrapper.h
Companion guide	Application File Management
Related sample code	File Wrappers with Core Data Documents FunHouse GLUT Quartz Composer WWDC 2005 TextEdit SimpleStickies

Overview

The `NSFileWrapper` class provides access to the attributes and contents of file-system nodes. A **file-system node** is a file, directory, or symbolic link. Instances of this class are known as **file wrappers**.

File wrappers represent a file-system node as an object that can be displayed as an image (and possibly edited in place), saved to the file system, or transmitted to another application.

There are three types of file wrappers:

- **Regular-file file wrapper:** Represents a regular file.
- **Directory file wrapper:** Represents a directory.
- **Symbolic-link file wrapper:** Represents a symbolic link.

A file wrapper has these attributes:

- **Filename.** Name of the file-system node the file wrapper represents.
- **file-system attributes.** See *NSFileManager Class Reference* for information on the contents of the `attributes` dictionary.
- **Regular-file contents.** Applicable only to regular-file file wrappers.
- **File wrappers.** Applicable only to directory file wrappers.

- **Destination node.** Applicable only to symbolic-link file wrappers.

Adopted Protocols

NSCoding
`encodeWithCoder:`
`initWithCoder:`

Tasks

Creating File Wrappers

This class has several designated initializers.

- `initWithURL:options:error:` (page 1127)
  Initializes a file wrapper instance whose kind is determined by the type of file-system node located by the URL.
- `initWithPath:` (page 1126)
  Initializes a file wrapper instance whose kind is determined by the type of file-system node located by the path. (Deprecated. Use `initWithURL:options:error:` (page 1127) instead.)
- `initDirectoryWithFileWrappers:` (page 1123)
  Initializes the receiver as a directory file wrapper, with a given file-wrapper list.
- `initRegularFileWithContents:` (page 1124)
  Initializes the receiver as a regular-file file wrapper.
- `initSymbolicLinkWithDestinationURL:` (page 1125)
  Initializes the receiver as a symbolic-link file wrapper that links to a specified file.
- `initWithSerializedRepresentation:` (page 1126)
  Initializes the receiver as a regular-file file wrapper from given serialized data.
- `initSymbolicLinkWithDestination:` (page 1124) Deprecated in Mac OS X v10.6
  Initializes the receiver as a symbolic-link file wrapper. (Deprecated. Use `initSymbolicLinkWithDestinationURL:` (page 1125) instead.)

Querying File Wrappers

- `isRegularFile` (page 1128)
  Indicates whether the receiver is a regular-file file wrapper.
- `isDirectory` (page 1128)
  Indicates whether the receiver is a directory file wrapper.
- `isSymbolicLink` (page 1128)
  Indicates whether the receiver is a symbolic-link file wrapper.

Accessing File-Wrapper Information

- `fileWrappers` (page 1122)
  Returns the file wrappers contained by a directory file wrapper.
- `addFileWrapper:` (page 1119)
  Adds a child file wrapper to the receiver, which must be a directory file wrapper.
- `removeFileWrapper:` (page 1133)
  Removes a child file wrapper from the receiver, which must be a directory file wrapper.
- `addRegularFileWithContents:preferredFilename:` (page 1120)
  Creates a regular-file file wrapper with the given contents and adds it to the receiver, which must be a directory file wrapper.
- `keyForFileWrapper:` (page 1129)
  Returns the dictionary key used by a directory to identify a given file wrapper.
- `symbolicLinkDestinationURL` (page 1136)
  Provides the URL referenced by the receiver, which must be a symbolic-link file wrapper.
- `addFileWithPath:` (page 1118) Deprecated in Mac OS X v10.6
  Creates a file wrapper from a given file-system node and adds it to the receiver, which must be a directory file wrapper. (Deprecated. Use `addFileWrapper:` (page 1119) instead.)
- `addSymbolicLinkWithDestination:preferredFilename:` (page 1121) Deprecated in Mac OS X v10.6
  Creates a symbolic-link file wrapper pointing to a given file-system node and adds it to the receiver, which must be a directory file wrapper. (Deprecated. Use `addFileWrapper:` (page 1119) instead.)
- `symbolicLinkDestination` (page 1136) Deprecated in Mac OS X v10.6
  Provides the pathname referenced by the receiver, which must be a symbolic-link file wrapper. (Deprecated. Use `symbolicLinkDestinationURL` (page 1136) instead.)

Updating File Wrappers

- `matchesContentsOfURL:` (page 1129)
  Indicates whether the contents of a file wrapper matches a directory, regular file, or symbolic link on disk.
- `readFromURL:options:error:` (page 1132)
  Recursively rereads the entire contents of a file wrapper from the specified location on disk.
- `needsToBeUpdatedFromPath:` (page 1130) Deprecated in Mac OS X v10.6
  Indicates whether the file wrapper needs to be updated to match a given file-system node. (Deprecated. Use `matchesContentsOfURL:` (page 1129) instead.)
- `updateFromPath:` (page 1137) Deprecated in Mac OS X v10.6
  Updates the file wrapper to match a given file-system node. (Deprecated. Use `readFromURL:options:error:` (page 1132) instead.)

Serializing

- `serializedRepresentation` (page 1133)
  Returns the contents of the file wrapper as an opaque collection of data.

Accessing Files

- `filename` (page 1122)
  Provides the filename of a file wrapper.
- `setFilename:` (page 1134)
  Specifies the filename of a file wrapper.
- `preferredFilename` (page 1131)
  Provides the preferred filename for a file wrapper.
- `setPreferredFilename:` (page 1135)
  Specifies the receiver's preferred filename.
- `fileAttributes` (page 1121)
  Returns a file wrapper's file attributes.
- `setFileAttributes:` (page 1134)
  Specifies a file wrapper's file attributes.
- `regularFileContents` (page 1132)
  Returns the contents of the file-system node associated with a regular-file file wrapper.

Writing Files

- `writeToURL:options:originalContentsURL:error:` (page 1138)
  Recursively writes the entire contents of a file wrapper to a given file-system URL.
- `writeToFile:atomically:updateFilenames:` (page 1137) Deprecated in Mac OS X v10.6
  Writes a file wrapper's contents to a given file-system node. (Deprecated. Use `writeToURL:options:originalContentsURL:error:` (page 1138) instead.)

Instance Methods

addFileWithPath:

Creates a file wrapper from a given file-system node and adds it to the receiver, which must be a directory file wrapper. (Deprecated in Mac OS X v10.6. Use `addFileWrapper:` (page 1119) instead.)

```
- (NSString *)addFileWithPath:(NSString *)node
```

Parameters

node

file-system node from which to create the file wrapper to add to the directory.

Return Value

Dictionary key used to store the new file wrapper in the directory's list of file wrappers. See "Working With Directory Wrappers" for more information.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Instead of using this method, you can instantiate `NSFileWrapper` with one of the initializers, send it `setPreferredFilename:` (page 1135) if necessary, and pass the result to `addFileWrapper:` (page 1119).

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `addRegularFileWithContents:preferredFilename:` (page 1120)
- `addSymbolicLinkWithDestination:preferredFilename:` (page 1121)
- `removeFileWrapper:` (page 1133)
- `fileWrappers` (page 1122)

Related Sample Code
File Wrappers with Core Data Documents

Declared In
`NSFileWrapper.h`

addFileWrapper:

Adds a child file wrapper to the receiver, which must be a directory file wrapper.

```
- (NSString *)addFileWrapper:(NSFileWrapper *)child
```

Parameters

child

File wrapper to add to the directory.

Return Value
Dictionary key used to store *fileWrapper* in the directory's list of file wrappers. The dictionary key is a unique filename, which is the same as the passed-in file wrapper's preferred filename unless that name is already in use as a key in the directory's dictionary of children. See "Working With Directory Wrappers" in *Application File Management* for more information about the file-wrapper list structure.

Discussion
Use this method to add an existing file wrapper as a child of a directory file wrapper. If the file wrapper does not have a preferred filename, use the `setPreferredFilename:` (page 1135) method to give it one before calling `addFileWrapper:`. To create a new file wrapper and add it to a directory, use the `addRegularFileWithContents:preferredFilename:` (page 1120) method.

Special Considerations

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

This method raises `NSInvalidArgumentException` if the child file wrapper doesn't have a preferred name.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addRegularFileWithContents:preferredFilename:` (page 1120)
- `removeFileWrapper:` (page 1133)
- `fileWrappers` (page 1122)
- `preferredFilename` (page 1131)

Related Sample Code
File Wrappers with Core Data Documents

Declared In
`NSFileWrapper.h`

addRegularFileWithContents:preferredFilename:

Creates a regular-file file wrapper with the given contents and adds it to the receiver, which must be a directory file wrapper.

```
- (NSString *)addRegularFileWithContents:(NSData *)data preferredFilename:(NSString
    *)filename
```

Parameters

data

Contents for the new regular-file file wrapper.

filename

Preferred filename for the new regular-file file wrapper.

Return Value
Dictionary key used to store the new file wrapper in the directory's list of file wrappers. The dictionary key is a unique filename, which is the same as the passed-in file wrapper's preferred filename unless that name is already in use as a key in the directory's dictionary of children. See "Working With Directory Wrappers" in *Application File Management* for more information about the file-wrapper list structure.

Discussion
This is a convenience method. The default implementation allocates a new file wrapper, initializes it with `initRegularFileWithContents:` (page 1124), sends it `setPreferredFilename:` (page 1135), adds it to the directory with `addFileWrapper:` (page 1119), and returns what `addFileWrapper:` returned.

Special Considerations

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

This method raises `NSInvalidArgumentException` if you pass `nil` or an empty value for `filename`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addFileWrapper:` (page 1119)
- `removeFileWrapper:` (page 1133)
- `fileWrappers` (page 1122)

Related Sample Code
FunHouse
SimpleStickies

Declared In
`NSFileWrapper.h`

addSymbolicLinkWithDestination:preferredFilename:

Creates a symbolic-link file wrapper pointing to a given file-system node and adds it to the receiver, which must be a directory file wrapper. (Deprecated in Mac OS X v10.6. Use `addFileWrapper:` (page 1119) instead.)

```
- (NSString *)addSymbolicLinkWithDestination:(NSString *)node
    preferredFilename:(NSString *)preferredFilename
```

Parameters

node

Pathname the new symbolic-link file wrapper is to reference.

preferredFilename

Preferred filename for the new symbolic-link file wrapper.

Return Value

Dictionary key used to store the new file wrapper in the directory's list of file wrappers. See "Working With Directory Wrappers" for more information.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Instead of using this method, you can instantiate `NSFileWrapper` with one of the initializers, send it `setPreferredFilename:` (page 1135) if necessary, and pass the result to `addFileWrapper:` (page 1119).

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

This method raises `NSInvalidArgumentException` if you pass `nil` or an empty value for `preferredFilename`.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- `addFileWrapper:` (page 1119)
- `addRegularFileWithContents:preferredFilename:` (page 1120)
- `removeFileWrapper:` (page 1133)
- `fileWrappers` (page 1122)

Declared In

`NSFileWrapper.h`

fileAttributes

Returns a file wrapper's file attributes.

```
- (NSDictionary *)fileAttributes
```

Return Value

File attributes, in a dictionary of the same sort as that returned by `attributesOfItemAtPath:error:` (`NSFileManager`).

Availability

Available in Mac OS X v10.0 and later.

See Also
- `setFileAttributes:` (page 1134)

Declared In
`NSFileWrapper.h`

filename

Provides the filename of a file wrapper.

```
- (NSString *)filename
```

Return Value
The file wrapper's filename; `nil` when the file wrapper has no corresponding file-system node.

Discussion
The filename is used for record-keeping purposes only and is set automatically when the file wrapper is created from the file system using `initWithURL:options:error:` (page 1127) and when it's saved to the file system using `writeToURL:options:originalContentsURL:error:` (page 1138) (although this method allows you to request that the filename not be updated).

The filename is usually the same as the preferred filename, but might instead be a name derived from the preferred filename. You can use this method to get the name of a child that's just been read. Don't use this method to get the name of a child that's about to be written, because the name might be about to change; send `keyForFileWrapper:` (page 1129) to the parent instead.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `preferredFilename` (page 1131)
- `setFilename:` (page 1134)

Related Sample Code
File Wrappers with Core Data Documents
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSFileWrapper.h`

fileWrappers

Returns the file wrappers contained by a directory file wrapper.

```
- (NSDictionary *)fileWrappers
```

Return Value
A key-value dictionary of the file wrappers contained in the directory. The dictionary contains entries whose values are the file wrappers and whose keys are the unique filenames that have been assigned to each one. See "Working With Directory Wrappers" in *Application File Management* for more information about the file-wrapper list structure.

Discussion
Returns a dictionary whose values are the file wrapper's children and whose keys are the unique filenames that have been assigned to each one. This method may return `nil` if the user modifies the directory after you call `readFromURL:options:error:` (page 1132) or `initWithURL:options:error:` (page 1127) but before `NSFileWrapper` has read the contents of the directory. Use the `NSFileWrapperReadingImmediate` reading option to reduce the likelihood of that problem.

Special Considerations

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `filename` (page 1122)
- `addFileWrapper:` (page 1119)

Related Sample Code
File Wrappers with Core Data Documents
FunHouse

Declared In
`NSFileWrapper.h`

initDirectoryWithFileWrappers:

Initializes the receiver as a directory file wrapper, with a given file-wrapper list.

```
- (id)initDirectoryWithFileWrappers:(NSDictionary *)childrenByPreferredName
```

Parameters

childrenByPreferredName

Key-value dictionary of file wrappers with which to initialize the receiver. The dictionary must contain entries whose values are the file wrappers that are to become children and whose keys are filenames. See "Working With Directory Wrappers" in *Application File Management* for more information about the file-wrapper list structure.

Return Value
Initialized file wrapper for *fileWrappers*.

Discussion
After initialization, the file wrapper is not associated with a file-system node until you save it using `writeToURL:options:originalContentsURL:error:` (page 1138).

The receiver is initialized with open permissions: anyone can read, write, or modify the directory on disk.

If any file wrapper in the directory doesn't have a preferred filename, its preferred name is automatically set to its corresponding key in the *childrenByPreferredName* dictionary.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPreferredFilename:` (page 1135)

- `filename` (page 1122)
- `setFileAttributes:` (page 1134)

Related Sample Code
File Wrappers with Core Data Documents
FunHouse
SimpleStickies

Declared In
`NSFileWrapper.h`

initRegularFileWithContents:

Initializes the receiver as a regular-file file wrapper.

```
- (id)initRegularFileWithContents:(NSData *)contents
```

Parameters
contents
Contents of the file.

Return Value
Initialized regular-file file wrapper containing *contents*.

Discussion
After initialization, the file wrapper is not associated with a file-system node until you save it using `writeToURL:options:originalContentsURL:error:` (page 1138).

The file wrapper is initialized with open permissions: anyone can write to or read the file wrapper. .

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPreferredFilename:` (page 1135)
- `filename` (page 1122)
- `fileAttributes` (page 1121)
- `regularFileContents` (page 1132)

Related Sample Code
File Wrappers with Core Data Documents
FunHouse

Declared In
`NSFileWrapper.h`

initSymbolicLinkWithDestination:

Initializes the receiver as a symbolic-link file wrapper. (Deprecated in Mac OS X v10.6. Use `initSymbolicLinkWithDestinationURL:` (page 1125) instead.)

```
- (id)initSymbolicLinkWithDestination:(NSString *)node
```

Parameters

node

Pathname the receiver is to represent.

Return Value

Initialized symbolic-link file wrapper referencing *node*.

Discussion

The receiver is not associated to a file-system node until you save it using `writeToFile:atomically:updateFilenames:` (page 1137). It's also initialized with open permissions; anyone can read or write the disk representations it saves.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Therefore, this method has been deprecated in favor of `initSymbolicLinkWithDestinationURL:` (page 1125).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- `setPreferredFilename:` (page 1135)
- `filename` (page 1122)
- `fileAttributes` (page 1121)

Declared In

`NSFileWrapper.h`

initSymbolicLinkWithDestinationURL:

Initializes the receiver as a symbolic-link file wrapper that links to a specified file.

```
- (id)initSymbolicLinkWithDestinationURL:(NSURL *)url
```

Parameters

url

URL of the file the file wrapper is to reference.

Return Value

Initialized symbolic-link file wrapper referencing *url*.

Discussion

The file wrapper is not associated with a file-system node until you save it using `writeToURL:options:originalContentsURL:error:` (page 1138).

The file wrapper is initialized with open permissions: anyone can modify or read the file reference. .

Availability

Available in Mac OS X v10.6 and later.

See Also

- `setPreferredFilename:` (page 1135)
- `filename` (page 1122)
- `fileAttributes` (page 1121)

Declared In
NSFileWrapper.h

initWithPath:

Initializes a file wrapper instance whose kind is determined by the type of file-system node located by the path. (Deprecated in Mac OS X v10.6. Use `initWithURL:options:error:` (page 1127) instead.)

```
- (id)initWithPath:(NSString *)node
```

Parameters

node

Pathname of the file-system node the file wrapper is to represent.

Return Value
File wrapper for *node*.

Discussion
If *node* is a directory, this method recursively creates file wrappers for each node within that directory.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Therefore, this method has been deprecated in favor of `initWithURL:options:error:` (page 1127).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `setPreferredFilename:` (page 1135)
- `filename` (page 1122)
- `fileAttributes` (page 1121)

Declared In
NSFileWrapper.h

initWithSerializedRepresentation:

Initializes the receiver as a regular-file file wrapper from given serialized data.

```
- (id)initWithSerializedRepresentation:(NSData *)serializedRepresentation
```

Parameters

serializedRepresentation

Serialized representation of a file wrapper in the format used for the `NSFileContentsPboardType` pasteboard type. Data of this format is returned by such methods as `serializedRepresentation` (page 1133) and `RTFDFromRange:documentAttributes:` (page 268) (`NSAttributedString`).

Return Value
Regular-file file wrapper initialized from *serializedRepresentation*.

Discussion
The file wrapper is not associated with a file-system node until you save it using `writeToURL:options:originalContentsURL:error:` (page 1138).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPreferredFilename:` (page 1135)
- `filename` (page 1122)
- `fileAttributes` (page 1121)

Declared In
`NSFileWrapper.h`

initWithURL:options:error:

Initializes a file wrapper instance whose kind is determined by the type of file-system node located by the URL.

```
- (id)initWithURL:(NSURL *)url options:(NSFileWrapperReadingOptions)options
    error:(NSError **)outError
```

Parameters

url

URL of the file-system node the file wrapper is to represent.

options

Option flags for reading the node located at `url`. See "File Wrapper Reading Options" (page 1139) for possible values.

outError

If an error occurs, upon return contains an `NSError` object that describes the problem. Pass `NULL` if you do not want error information.

Return Value
File wrapper for the file-system node at *url*. May be a directory, file, or symbolic link, depending on what is located at the URL. Returns `NO` (0) if reading is not successful.

Discussion
If *url* is a directory, this method recursively creates file wrappers for each node within that directory. Use the `fileWrappers` (page 1122) method to get the file wrappers of the nodes contained by the directory.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `fileWrappers` (page 1122)
- `setPreferredFilename:` (page 1135)
- `filename` (page 1122)
- `fileAttributes` (page 1121)
- `readFromURL:options:error:` (page 1132)

Declared In
NSFileWrapper.h

isDirectory

Indicates whether the receiver is a directory file wrapper.

```
- (BOOL)isDirectory
```

Return Value
YES when the receiver is a directory file wrapper, NO otherwise.

Discussion
Invocations of readFromURL:options:error: (page 1132) may change what is returned by subsequent invocations of this method if the type of the file on disk has changed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isRegularFile (page 1128)
- isSymbolicLink (page 1128)

Declared In
NSFileWrapper.h

isRegularFile

Indicates whether the receiver is a regular-file file wrapper.

```
- (BOOL)isRegularFile
```

Return Value
YES when the receiver is a regular-file wrapper, NO otherwise.

Discussion
Invocations of readFromURL:options:error: (page 1132) may change what is returned by subsequent invocations of this method if the type of the file on disk has changed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isDirectory (page 1128)
- isSymbolicLink (page 1128)

Declared In
NSFileWrapper.h

isSymbolicLink

Indicates whether the receiver is a symbolic-link file wrapper.

```
- (BOOL)isSymbolicLink
```

Return Value
`YES` when the receiver is a symbolic-link file wrapper, `NO` otherwise.

Discussion
Invocations of `readFromURL:options:error:` (page 1132) may change what is returned by subsequent invocations of this method if the type of the file on disk has changed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isDirectory` (page 1128)
- `isRegularFile` (page 1128)

Declared In
`NSFileWrapper.h`

keyForFileWrapper:

Returns the dictionary key used by a directory to identify a given file wrapper.

```
- (NSString *)keyForFileWrapper:(NSFileWrapper *)child
```

Parameters
child
The child file wrapper for which you want the key.

Return Value
Dictionary key used to store the file wrapper in the directory's list of file wrappers. The dictionary key is a unique filename, which may not be the same as the passed-in file wrapper's preferred filename if more than one file wrapper in the directory's dictionary of children has the same preferred filename. See "Working With Directory Wrappers" in *Application File Management* for more information about the file-wrapper list structure. Returns `nil` if the file wrapper specified in `child` is not a child of the directory.

Special Considerations
This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `filename` (page 1122)

Declared In
`NSFileWrapper.h`

matchesContentsOfURL:

Indicates whether the contents of a file wrapper matches a directory, regular file, or symbolic link on disk.

```
- (BOOL)matchesContentsOfURL:(NSURL *)url
```

Parameters

url

URL of the file-system node with which to compare the file wrapper.

Return Value

YES when the contents of the file wrapper match the contents of *url*, NO otherwise.

Discussion

The contents of files are not compared; matching of regular files is based on file modification dates. For a directory, children are compared against the files in the directory, recursively.

Because children of directory file wrappers are not read immediately by the `initWithURL:options:error:` (page 1127) method unless the `NSFileWrapperReadingImmediate` reading option is used, even a newly-created directory file wrapper might not have the same contents as the directory on disk. You can use this method to determine whether the file wrapper's contents in memory need to be updated.

If the file wrapper needs updating, use the `readFromURL:options:error:` (page 1132) method with the `NSFileWrapperReadingImmediate` reading option.

This table describes which attributes of the file wrapper and file-system node are compared to determine whether the file wrapper matches the node on disk:

File-wrapper type	Comparison determinants
Regular file	Modification date and access permissions.
Directory	Children (recursive).
Symbolic link	Destination pathname.

Availability

Available in Mac OS X v10.6 and later.

See Also

- `readFromURL:options:error:` (page 1132)
- `fileAttributes` (page 1121)

Declared In

NSFileWrapper.h

needsToBeUpdatedFromPath:

Indicates whether the file wrapper needs to be updated to match a given file-system node. (Deprecated in Mac OS X v10.6. Use `matchesContentsOfURL:` (page 1129) instead.)

```
- (BOOL)needsToBeUpdatedFromPath:(NSString *)node
```

Parameters

node

file-system node with which to compare the file wrapper.

Return Value

YES when the file wrapper needs to be updated to match *node*, NO otherwise.

Discussion
This table describes which attributes of the file wrapper and *node* are compared to determine whether the file wrapper needs to be updated:

File-wrapper type	Comparison determinants
Regular file	Modification date and access permissions.
Directory	Member hierarchy (recursive).
Symbolic link	Destination pathname.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Therefore, this method has been deprecated in favor of `matchesContentsOfURL:` (page 1129).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `updateFromPath:` (page 1137)
- `fileAttributes` (page 1121)

Declared In
`NSFileWrapper.h`

preferredFilename

Provides the preferred filename for a file wrapper.

```
- (NSString *)preferredFilename
```

Return Value
The file wrapper's preferred filename.

Discussion
This name is normally used as the dictionary key when a child file wrapper is added to a directory file wrapper. However, if another file wrapper with the same preferred name already exists in the directory file wrapper when the receiver is added, the filename assigned as the dictionary key may differ from the preferred filename.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `filename` (page 1122)
- `setPreferredFilename:` (page 1135)

Declared In
`NSFileWrapper.h`

readFromURL:options:error:

Recursively rereads the entire contents of a file wrapper from the specified location on disk.

```
- (BOOL)readFromURL:(NSURL *)url options:(NSFileWrapperReadingOptions)options
    error:(NSError **)outError
```

Parameters

url

URL of the file-system node corresponding to the file wrapper.

options

Option flags for reading the node located at `url`. See "File Wrapper Reading Options" (page 1139) for possible values.

outError

If an error occurs, upon return contains an `NSError` object that describes the problem. Pass `NULL` if you do not want error information.

Return Value

`YES` if successful. If not successful, returns `NO` after setting `outError` to an `NSError` object that describes the reason why the file wrapper could not be reread.

Discussion

When reading a directory, children are added and removed as necessary to match the file system.

Availability

Available in Mac OS X v10.6 and later.

See Also

- `initWithURL:options:error:` (page 1127)
- `fileWrappers` (page 1122)
- `filename` (page 1122)
- `fileAttributes` (page 1121)
- `writeToURL:options:originalContentsURL:error:` (page 1138)

Declared In

`NSFileWrapper.h`

regularFileContents

Returns the contents of the file-system node associated with a regular-file file wrapper.

```
- (NSData *)regularFileContents
```

Return Value

Contents of the file-system node the file wrapper represents.

Discussion

This method may return `nil` if the user modifies the file after you call `readFromURL:options:error:` (page 1132) or `initWithURL:options:error:` (page 1127) but before `NSFileWrapper` has read the contents of the file. Use the `NSFileWrapperReadingImmediate` reading option to reduce the likelihood of that problem.

Special Considerations

This method raises `NSInternalInconsistencyException` if the receiver is not a regular-file file wrapper.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initRegularFileWithContents:` (page 1124)
- `readFromURL:options:error:` (page 1132)

Related Sample Code
File Wrappers with Core Data Documents
FunHouse
SimpleStickies

Declared In
`NSFileWrapper.h`

removeFileWrapper:

Removes a child file wrapper from the receiver, which must be a directory file wrapper.

```
- (void)removeFileWrapper:(NSFileWrapper *)child
```

Parameters
child
File wrapper to remove from the directory.

Special Considerations
This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addFileWrapper:` (page 1119)
- `addRegularFileWithContents:preferredFilename:` (page 1120)
- `fileWrappers` (page 1122)

Related Sample Code
File Wrappers with Core Data Documents

Declared In
`NSFileWrapper.h`

serializedRepresentation

Returns the contents of the file wrapper as an opaque collection of data.

```
- (NSData *)serializedRepresentation
```

Return Value
The file wrapper's contents in the format used for the pasteboard type `NSFileContentsPboardType`.

Discussion
Returns an `NSData` object suitable for passing to `initWithSerializedRepresentation:` (page 1126). This method may return `nil` if the user modifies the contents of the file-system node after you call `readFromURL:options:error:` (page 1132) or `initWithURL:options:error:` (page 1127) but before `NSFileWrapper` has read the contents of the file. Use the `NSFileWrapperReadingImmediate` reading option to reduce the likelihood of that problem.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- initWithSerializedRepresentation:` (page 1126)

Declared In
`NSFileWrapper.h`

setFileAttributes:

Specifies a file wrapper's file attributes.

```
- (void)setFileAttributes:(NSDictionary *)fileAttributes
```

Parameters
fileAttributes
File attributes for the file wrapper, in a dictionary of the same sort as that used by `setAttributes:ofItemAtPath:error:` (`NSFileManager`).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- fileAttributes` (page 1121)
`- writeToURL:options:originalContentsURL:error:` (page 1138)

Declared In
`NSFileWrapper.h`

setFilename:

Specifies the filename of a file wrapper.

```
- (void)setFilename:(NSString *)filename
```

Parameters
filename
Filename of the file wrapper.

Discussion
The file name is a dictionary key used to store *fileWrapper* in a directory's list of child file wrappers. The dictionary key is a unique filename, which is the same as the child file wrapper's preferred filename unless that name is already in use as a key in the directory's dictionary of children. See "Working With Directory Wrappers" in *Application File Management* for more information about the file-wrapper list structure. In general, the filename is set for you by the `initWithURL:options:error:` (page 1127),

`initDirectoryWithFileWrappers:` (page 1123), or `writeToURL:options:originalContentsURL:error:` (page 1138) methods; you do not normally have to call this method directly.

Special Considerations

This method raises `NSInvalidArgumentException` if you pass `nil` or an empty value for `filename`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `filename` (page 1122)
- `setPreferredFilename:` (page 1135)
- `initWithURL:options:error:` (page 1127)
- `initDirectoryWithFileWrappers:` (page 1123)
- `writeToURL:options:originalContentsURL:error:` (page 1138)

Declared In
`NSFileWrapper.h`

setPreferredFilename:

Specifies the receiver's preferred filename.

```
- (void)setPreferredFilename:(NSString *)filename
```

Parameters

filename
Preferred filename for the receiver.

Discussion
When a file wrapper is added to a parent directory file wrapper, the parent attempts to use the child's preferred filename as the key in its dictionary of children. If that key is already in use, then the parent derives a unique filename from the preferred filename and uses that for the key.

When you change the preferred filename of a file wrapper, the default implementation of this method causes existing parent directory file wrappers to remove and re-add the child to accommodate the change. Preferred filenames of children are not preserved when you write a file wrapper to disk and then later instantiate another file wrapper by reading the file from disk. If you need to preserve the user-visible names of attachments, you have to store the names yourself.

Special Considerations

This method raises `NSInvalidArgumentException` if you pass `nil` or an empty value for `filename`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `preferredFilename` (page 1131)
- `setFilename:` (page 1134)
- `addFileWrapper:` (page 1119)
- `initWithURL:options:error:` (page 1127)

- initDirectoryWithFileWrappers: (page 1123)
- writeToURL:options:originalContentsURL:error: (page 1138)

Related Sample Code
File Wrappers with Core Data Documents
GLUT

Declared In
NSFileWrapper.h

symbolicLinkDestination

Provides the pathname referenced by the receiver, which must be a symbolic-link file wrapper. (Deprecated in Mac OS X v10.6. Use symbolicLinkDestinationURL (page 1136) instead.)

```
- (NSString *)symbolicLinkDestination
```

Return Value
Pathname the file wrapper references (the destination of the symbolic link the file wrapper represents).

Special Considerations
Beginning with Mac OS X v10.6, the preferred method of referring to files is with a file:// URL. Therefore, this method has been deprecated in favor of symbolicLinkDestinationURL (page 1136).

This method raises NSInternalInconsistencyException if the receiver is not a symbolic-link file wrapper.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

Declared In
NSFileWrapper.h

symbolicLinkDestinationURL

Provides the URL referenced by the receiver, which must be a symbolic-link file wrapper.

```
- (NSURL *)symbolicLinkDestinationURL
```

Return Value
Pathname the file wrapper references (that is, the destination of the symbolic link the file wrapper represents).

Discussion
This method may return nil if the user modifies the symbolic link after you call readFromURL:options:error: (page 1132) or initWithURL:options:error: (page 1127) but before NSFileWrapper has read the contents of the link. Use the NSFileWrapperReadingImmediate reading option to reduce the likelihood of that problem.

Special Considerations
This method raises NSInternalInconsistencyException if the receiver is not a symbolic-link file wrapper.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSFileWrapper.h`

updateFromPath:

Updates the file wrapper to match a given file-system node. (Deprecated in Mac OS X v10.6. Use `readFromURL:options:error:` (page 1132) instead.)

```
- (BOOL)updateFromPath:(NSString *)path
```

Return Value
`YES` if the update is carried out, `NO` if it isn't needed.

Discussion
For a directory file wrapper, the contained file wrappers are also sent `updateFromPath:` messages. If nodes in the corresponding directory on the file system have been added or removed, corresponding file wrappers are released or created as needed.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Therefore, this method has been deprecated in favor of `readFromURL:options:error:` (page 1132).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
`- needsToBeUpdatedFromPath:` (page 1130)
`updateAttachmentsFromPath:` (page 1714) (NSAttributedString)

Declared In
`NSFileWrapper.h`

writeToFile:atomically:updateFilenames:

Writes a file wrapper's contents to a given file-system node. (Deprecated in Mac OS X v10.6. Use `writeToURL:options:originalContentsURL:error:` (page 1138) instead.)

```
- (BOOL)writeToFile:(NSString *)node atomically:(BOOL)atomically
    updateFilenames:(BOOL)updateNames
```

Parameters
node
Pathname of the file-system node to which the receiver's contents are written.

atomically

YES to write the file safely so that:

- An existing file is not overwritten
- The method fails if the file cannot be written in its entirety

NO to overwrite an existing file and ignore incomplete writes.

updateNames

YES to update the receiver's filenames (its filename and—for directory file wrappers—the filenames of its sub–file wrappers) be changed to the filenames of the corresponding nodes in the file system, after a successful write operation. Use this in Save or Save As operations.

NO to specify that the receiver's filenames not be updated. Use this in Save To operations.

Return Value

YES when the write operation is successful, NO otherwise.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Therefore, this method has been deprecated in favor of `writeToURL:options:originalContentsURL:error:` (page 1138).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- `filename` (page 1122)
- `writeToURL:options:originalContentsURL:error:` (page 1138)

Related Sample Code

File Wrappers with Core Data Documents

Quartz Composer WWDC 2005 TextEdit

Declared In

NSFileWrapper.h

writeToURL:options:originalContentsURL:error:

Recursively writes the entire contents of a file wrapper to a given file-system URL.

```
- (BOOL)writeToURL:(NSURL *)url options:(NSFileWrapperWritingOptions)options
    originalContentsURL:(NSURL *)originalContentsURL error:(NSError **)outError
```

Parameters

url

URL of the file-system node to which the file wrapper's contents are written.

options

Option flags for writing to the node located at `url`. See "File Wrapper Writing Options" (page 1140) for possible values.

originalContentsURL

The location of a previous revision of the contents being written. The default implementation of this method attempts to avoid unnecessary I/O by writing hard links to regular files instead of actually writing out their contents when the contents have not changed. The child file wrappers must return accurate values when sent the `filename` (page 1122) method for this to work. Use the `NSFileWrapperWritingWithNameUpdating` writing option to increase the likelihood of that.

Specify `nil` for this parameter if there is no earlier version of the contents or if you want to ensure that all the contents are written to files.

updateNames

`YES` to update the receiver's filenames (its filename and—for directory file wrappers—the filenames of its sub–file wrappers) be changed to the filenames of the corresponding nodes in the file system, after a successful write operation. Use this in Save or Save As operations.

`NO` to specify that the receiver's filenames not be updated. Use this in Save To operations.

outError

If an error occurs, upon return contains an `NSError` object that describes the problem. Pass `NULL` if you do not want error information.

Return Value

`YES` when the write operation is successful. If not successful, returns `NO` after setting `outError` to an `NSError` object that describes the reason why the file wrapper's contents could not be written.

Availability

Available in Mac OS X v10.6 and later.

See Also

- `filename` (page 1122)
- `readFromURL:options:error:` (page 1132)

Declared In

`NSFileWrapper.h`

Constants

File Wrapper Reading Options

Reading options that can be set by the `initWithURL:options:error:` (page 1127) and `readFromURL:options:error:` (page 1132) methods.

```
enum {
 NSFileWrapperReadingImmediate = 1 << 0,
 NSFileWrapperReadingWithoutMapping = 1 << 1
};
typedef NSUInteger NSFileWrapperReadingOptions;
```

Constants

`NSFileWrapperReadingImmediate`

If reading with this option succeeds, then subsequent invocations of `fileWrappers` (page 1122), `regularFileContents` (page 1132), `symbolicLinkDestinationURL` (page 1136), and `serializedRepresentation` (page 1133) sent to the file wrapper and all its child file wrappers will fail and return `nil` only if an actual error occurs (for example, the volume has disappeared or the file server is unreachable)—not as a result of the user moving or deleting files.

For performance reasons, `NSFileWrapper` may not read the contents of some file packages immediately even when this option is chosen. For example, because the contents of bundles (not all file packages are bundles) are immutable to the user, `NSFileWrapper` may read the children of such a directory lazily.

You can use this option to take a snapshot of a file or folder for writing later. For example, an application like TextEdit can use this option when creating new file wrappers to represent attachments that the user creates by copying and pasting or dragging and dropping from the Finder to a TextEdit document. Don't use this option when reading a document file package, because that would cause unnecessarily bad performance. For example, an application wouldn't use this option when creating file wrappers to represent attachments as it's opening a document stored in a file package.

Available in Mac OS X v10.6 and later.

Declared in `NSFileWrapper.h`.

`NSFileWrapperReadingWithoutMapping`

Whether file mapping for regular file wrappers is disallowed.

You can use this option to keep `NSFileWrapper` from memory-mapping files. This is useful if you want to make sure your application doesn't hold files open (mapped files are open files), therefore preventing the user from ejecting DVDs, unmounting disk partitions, or unmounting disk images. In Mac OS X v10.6 and later, `NSFileWrapper` memory-maps files that are on internal drives only. It never memory-maps files on external drives or network volumes, regardless of whether this option is used.

Available in Mac OS X v10.6 and later.

Declared in `NSFileWrapper.h`.

Discussion

You can use the `NSFileWrapperReadingImmediate` and `NSFileWrapperReadingWithoutMapping` reading options together to take an exact snapshot of a file-system hierarchy that is safe from all errors (including the ones mentioned above) once reading has succeeded. If reading with both options succeeds, then subsequent invocations of the methods listed in the comment for the `NSFileWrapperReadingImmediate` reading option to the receiver and all its descendant file wrappers will never fail. However, note that reading with both options together is expensive in terms of both I/O and memory for large files, or directories containing large files, or even directories containing many small files.

File Wrapper Writing Options

Writing options that can be set by the `writeToURL:options:originalContentsURL:error:` (page 1138) method.

```
enum {
 NSFileWrapperWritingAtomic = 1 << 0,
 NSFileWrapperWritingWithNameUpdating = 1 << 1
};
typedef NSUInteger NSFileWrapperWritingOptions;
```

Constants

`NSFileWrapperWritingAtomic`

Whether writing is done atomically.

You can use this option to ensure that, when overwriting a file package, the overwriting either completely succeeds or completely fails, with no possibility of leaving the file package in an inconsistent state. Because this option causes additional I/O, you shouldn't use it unnecessarily. For example, don't use this option in an override of `-[NSDocument writeToURL:ofType:error:]` (page 996), because `NSDocument` safe-saving is already done atomically.

Available in Mac OS X v10.6 and later.

Declared in `NSFileWrapper.h`.

`NSFileWrapperWritingWithNameUpdating`

Whether descendant file wrappers are sent the `setFilename:` (page 1134) method if the writing succeeds.

This option is necessary when your application passes a URL in the `originalContentsURL` parameter to the `writeToURL:options:originalContentsURL:error:` (page 1138) method. Without using this option (and reusing child file wrappers properly), subsequent invocations of `writeToURL:options:originalContentsURL:error:` (page 1138) would not be able to reliably create hard links in a new file package, because the record of names in the old file package would be out of date.

Available in Mac OS X v10.6 and later.

Declared in `NSFileWrapper.h`.

NSFont Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSFont.h
Companion guide	Font Handling
Related sample code	CoreTextArcCocoa FunHouse GeekGameBoard IBFragmentView Quartz Composer WWDC 2005 TextEdit

Overview

`NSFont` objects represent fonts to an application, providing access to characteristics of the font and assistance in laying out glyphs relative to one another. Font objects are also used to establish the current font for drawing text directly into a graphics context, using the `set` (page 1179) method.

You don't create `NSFont` objects using the `alloc` and `init` methods. Instead, you use either `fontWithDescriptor:size:` (page 1150) or `fontWithName:size:` (page 1152) to look up an available font and alter its size or matrix to your needs. These methods check for an existing font object with the specified characteristics, returning it if there is one. Otherwise, they look up the font data requested and create the appropriate object. `NSFont` also defines a number of methods for getting standard system fonts, such as `systemFontOfSize:` (page 1157), `userFontOfSize:` (page 1161), and `messageFontOfSize:` (page 1154). To request the default size for these standard fonts, pass a negative number or 0 as the font size.

Adopted Protocols

NSCoding

- `encodeWithCoder:`
- `initWithCoder:`

NSCopying
- copyWithZone:

Tasks

Creating Arbitrary Fonts

+ fontWithName:size: (page 1152)
Creates a font object for the specified font name and font size.

+ fontWithDescriptor:size: (page 1150)
Returns a font object for the specified font descriptor and font size.

+ fontWithDescriptor:textTransform: (page 1150)
Returns a font object for the specified font descriptor and text transform.

+ fontWithName:matrix: (page 1151)
Returns a font object for the specified font name and matrix.

Creating User Fonts

+ userFontOfSize: (page 1161)
Returns the font used by default for documents and other text under the user's control (that is, text whose font the user can normally change), in the specified size.

+ userFixedPitchFontOfSize: (page 1161)
Returns the font used by default for documents and other text under the user's control (that is, text whose font the user can normally change), when that font should be fixed-pitch, in the specified size.

Creating System Fonts

+ boldSystemFontOfSize: (page 1149)
Returns the Aqua system font used for standard interface items that are rendered in boldface type in the specified size.

+ controlContentFontOfSize: (page 1149)
Returns the font used for the content of controls in the specified size.

+ labelFontOfSize: (page 1152)
Returns the Aqua font used for standard interface labels in the specified size.

+ menuFontOfSize: (page 1154)
Returns the font used for menu items, in the specified size.

+ menuBarFontOfSize: (page 1153)
Returns the font used for menu bar items, in the specified size.

+ messageFontOfSize: (page 1154)
Returns the font used for standard interface items, such as button labels, menu items, and so on, in the specified size.

+ `paletteFontOfSize:` (page 1155)
  Returns the font used for palette window title bars, in the specified size.

+ `systemFontOfSize:` (page 1157)
  Returns the Aqua system font used for standard interface items, such as button labels, menu items, and so on, in the specified size.

+ `titleBarFontOfSize:` (page 1159)
  Returns the font used for window title bars, in the specified size.

+ `toolTipsFontOfSize:` (page 1160)
  Returns the font used for tool tips labels, in the specified size.

Using a Font to Draw

- `set` (page 1179)
  Sets this font as the font for the current graphics context.

- `setInContext:` (page 1180)
  Sets this font as the font for the specified graphics context.

Getting General Font Information

- `coveredCharacterSet` (page 1165)
  Returns an `NSCharacterSet` object containing all of the nominal characters renderable by the receiver, which is all of the entries mapped in the receiver's '`cmap`' table.

- `fontDescriptor` (page 1167)
  Returns the receiver's font descriptor.

- `isFixedPitch` (page 1171)
  Returns a Boolean value indicating whether all glyphs in the receiver have the same advancement.

- `mostCompatibleStringEncoding` (page 1173)
  Returns the string encoding that works best with the receiver, where there are the fewest possible unmatched characters in the string encoding and glyphs in the font.

- `renderingMode` (page 1178)
  Returns the rendering mode of the receiver.

Getting Information About Glyphs

- `glyphWithName:` (page 1170)
  Returns the named encoded glyph, or –1 if the receiver contains no such glyph.

Getting Metrics Information

+ `labelFontSize` (page 1153)
  Returns the size of the standard label font.

+ `smallSystemFontSize` (page 1157)
  Returns the size of the standard small system font.

+ `systemFontSize` (page 1158)
  Returns the size of the standard system font.

+ `systemFontSizeForControlSize:` (page 1159)
  Returns the font size used for the specified control size.

- `advancementForGlyph:` (page 1162)
  Returns the nominal spacing for the given glyph—the distance the current point moves after showing the glyph—accounting for the receiver's size.

- `ascender` (page 1163)
  Returns the top y-coordinate, offset from the baseline, of the receiver's longest ascender.

- `boundingRectForFont` (page 1163)
  Returns the receiver's bounding rectangle, scaled to the font's size.

- `boundingRectForGlyph:` (page 1164)
  Returns the bounding rectangle for the specified glyph, scaled to the receiver's size.

- `capHeight` (page 1164)
  Returns the receiver's cap height.

- `descender` (page 1166)
  Returns the bottom y coordinate, offset from the baseline, of the receiver's longest descender.

- `getAdvancements:forGlyphs:count:` (page 1168)
  Returns an array of the advancements for the specified glyphs rendered by the receiver.

- `getAdvancements:forPackedGlyphs:length:` (page 1168)
  Returns an array of the advancements for the specified packed glyphs and rendered by the receiver.

- `getBoundingRects:forGlyphs:count:` (page 1169)
  Returns an array of the bounding rectangles for the specified glyphs rendered by the receiver.

- `italicAngle` (page 1171)
  Returns the receiver's italic angle, the amount that the font is slanted in degrees counterclockwise from the vertical, as read from its AFM file. Because the slant is measured counterclockwise, English italic fonts typically return a negative value.

- `leading` (page 1172)
  Returns the receiver's leading.

- `matrix` (page 1172)
  Returns the receiver's font matrix, a standard six-element transformation matrix as used in the PostScript language, specifically with the `makefont` operator.

- `maximumAdvancement` (page 1172)
  Returns the greatest advancement of any of the receiver's glyphs.

- `numberOfGlyphs` (page 1173)
  Returns the number of glyphs in the receiver.

- `pointSize` (page 1174)
  Returns the receiver's point size, or the effective vertical point size for a font with a nonstandard matrix.

- `textTransform` (page 1180)
  Returns the current transformation matrix for the receiver.

- `underlinePosition` (page 1181)
  Returns the baseline offset that should be used when drawing underlines with the receiver, as determined by the font's AFM file.

- `underlineThickness` (page 1181)
  Returns the thickness that should be used when drawing underlines with the receiver, as determined by the font's AFM file.
- `xHeight` (page 1182)
  Returns the x-height of the receiver.

Getting Font Names

- `displayName` (page 1166)
  Returns the name, including family and face, used to represent the font in the user interface, typically localized for the user's language.
- `familyName` (page 1167)
  Returns the receiver's family name—for example, "Times" or "Helvetica."
- `fontName` (page 1167)
  Returns the receiver's full font name, as used in PostScript language code—for example, "Times-Roman" or "Helvetica-Oblique."

Setting User Fonts

+ `setUserFont:` (page 1157)
  Sets the font used by default for documents and other text under the user's control to the specified font.
+ `setUserFixedPitchFont:` (page 1156)
  Sets the font used by default for documents and other text under the user's control, when that font should be fixed-pitch, to the specified font.

Getting Corresponding Device Fonts

- `printerFont` (page 1178)
  Returns the scalable PostScript font corresponding to itself.
- `screenFont` (page 1178)
  Returns the bitmapped screen font corresponding to itself.
- `screenFontWithRenderingMode:` (page 1179)
  Returns a bitmapped screen font, when sent to a font object representing a scalable PostScript font, with the specified rendering mode, matching the receiver in typeface and matrix (or size), or `nil` if such a font can't be found.

Deprecated Methods

+ `preferredFontNames` (page 1155) Deprecated in Mac OS X v10.4
  Returns the names of fonts that the Application Kit tries first when a character has no font specified. (Deprecated. The `NSFontDescriptor` constant `NSFontCascadeListAttribute` offers more powerful font substitution management.)

`+ setPreferredFontNames:` (page 1156) Deprecated in Mac OS X v10.4

Sets the list of preferred font names. (Deprecated. The `NSFontDescriptor` constant `NSFontCascadeListAttribute` offers more powerful font substitution management.)

`+ useFont:` (page 1160) Deprecated in Mac OS X v10.4

Records the given font name as one used in the current print operation. (Deprecated. This is now automatically handled by Quartz.)

`- afmDictionary` (page 1163) Deprecated in Mac OS X v10.4

Returns the AFM font's dictionary. (Deprecated. Use accessor functions listed in "`Keys to the AFM Dictionary`" (page 1184) instead.)

`- defaultLineHeightForFont` (page 1165) Deprecated in Mac OS X v10.4

Returns the default line height for the receiver. (Deprecated. Use the `NSLayoutManager` method `defaultLineHeightForFont:` (page 1456) instead.)

`- encodingScheme` (page 1166) Deprecated in Mac OS X v10.4

Returns the name of the receiver's encoding scheme. (Deprecated. Use `mostCompatibleStringEncoding` (page 1173) instead.)

`- glyphIsEncoded:` (page 1169) Deprecated in Mac OS X v10.4

Returns a Boolean value indicating whether the receiver encodes the given glyph. (Deprecated. The value can be deduced by `aGlyph < [NSFont numberOfGlyphs]` since only `NSNativeShortGlyphPacking` is supported.)

`- glyphPacking` (page 1170) Deprecated in Mac OS X v10.4

Returns the best way to encode the receiver's glyphs into an array of bytes. (Deprecated. Only `NSNativeShortGlyphPacking` (page 1184) is supported.)

`- isBaseFont` (page 1170) Deprecated in Mac OS X v10.4

Returns a Boolean value indicating whether the receiver is a PostScript base font. (Deprecated. This information is not relevant to Mac OS X.)

`- positionOfGlyph:forCharacter:struckOverRect:` (page 1174) Deprecated in Mac OS X v10.4

Calculates and returns a suitable location for the given glyph to be drawn. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- positionOfGlyph:precededByGlyph:isNominal:` (page 1175) Deprecated in Mac OS X v10.4

Calculates and returns the location of a glyph. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- positionOfGlyph:struckOverGlyph:metricsExist:` (page 1175) Deprecated in Mac OS X v10.4

Calculates and returns a suitable location for the given glyph to be drawn. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- positionOfGlyph:struckOverRect:metricsExist:` (page 1176) Deprecated in Mac OS X v10.4

Overridden by subclasses to calculate and return a suitable location for a glyph to be drawn. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- positionOfGlyph:withRelation:toBaseGlyph:totalAdvancement:metricsExist:` (page 1176) Deprecated in Mac OS X v10.4

Calculates and returns a suitable location for a glyph to be drawn. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- positionsForCompositeSequence:numberOfGlyphs:pointArray:` (page 1177) Deprecated in Mac OS X v10.4

Calculates glyph locations. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

`- widthOfString:` (page 1181) Deprecated in Mac OS X v10.4
Returns the x-axis offset of the current point when the specified string is drawn with a `show` operator in the receiving font. (Deprecated. Use the Application Kit string-drawing methods, as described in NSString Additions (page 2581).)

Class Methods

boldSystemFontOfSize:

Returns the Aqua system font used for standard interface items that are rendered in boldface type in the specified size.

```
+ (NSFont *)boldSystemFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize
The size in points to which the font is scaled.

Return Value
A font object of the specified size.

Discussion
If *fontSize* is 0 or negative, returns the boldface system font at the default size.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ fontWithName:size:` (page 1152)

Related Sample Code
ImageBrowserViewAppearance
ImageKitDemo
PDF Calendar
PDFKitLinker2
Worm

Declared In
`NSFont.h`

controlContentFontOfSize:

Returns the font used for the content of controls in the specified size.

```
+ (NSFont *)controlContentFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize
The size in points to which the font is scaled.

Return Value
A font object of the specified size.

Discussion
For example, in a table, the user's input uses the control content font, and the table's header uses another font. If *fontSize* is 0 or negative, returns the control content font at the default size.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ fontWithName:size: (page 1152)

Declared In
NSFont.h

fontWithDescriptor:size:

Returns a font object for the specified font descriptor and font size.

```
+ (NSFont *)fontWithDescriptor:(NSFontDescriptor *)fontDescriptor
    size:(CGFloat)fontSize
```

Parameters

fontDescriptor
A font descriptor object.

fontSize
The size in points to which the font is scaled.

Return Value
A font object for the specified descriptor and size.

Discussion
In most cases, you can simply use fontWithName:size: (page 1152) to create standard scaled fonts.

Availability
Available in Mac OS X v10.4 and later.

See Also
+ fontWithName:size: (page 1152)

Declared In
NSFont.h

fontWithDescriptor:textTransform:

Returns a font object for the specified font descriptor and text transform.

```
+ (NSFont *)fontWithDescriptor:(NSFontDescriptor *)fontDescriptor
    textTransform:(NSAffineTransform *)textTransform
```

Parameters

fontDescriptor

The font descriptor object describing the font to return.

textTransform

An affine transformation applied to the font.

Return Value
A font object for the specified name and transform.

Discussion
In most cases, you can simply use `fontWithName:size:` (page 1152) to create standard scaled fonts. If *textTransform* is non-nil, it has precedence over `NSFontMatrixAttribute` in *fontDescriptor*.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ fontWithName:size:` (page 1152)

Declared In
`NSFont.h`

fontWithName:matrix:

Returns a font object for the specified font name and matrix.

```
+ (NSFont *)fontWithName:(NSString *)fontName matrix:(const CGFloat *)fontMatrix
```

Parameters

fontName

The fully specified family-face name of the font.

fontMatrix

A transformation matrix applied to the font.

Return Value
A font object for the specified name and transformation matrix.

Discussion
The *fontName* is a fully specified family-face name, such as Helvetica-BoldOblique or Times-Roman (not a name as shown in the Font Panel). The *fontMatrix* is a standard 6-element transformation matrix as used in the PostScript language, specifically with the `makefont` operator. In most cases, you can simply use `fontWithName:size:` (page 1152) to create standard scaled fonts.

You can use the defined value `NSFontIdentityMatrix` for [1 0 0 1 0 0]. Fonts created with a matrix other than `NSFontIdentityMatrix` don't automatically flip themselves in flipped views.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- isFlipped` (page 3181) (`NSView`)

Declared In
`NSFont.h`

fontWithName:size:

Creates a font object for the specified font name and font size.

```
+ (NSFont *)fontWithName:(NSString *)fontName size:(CGFloat)fontSize
```

Parameters

fontName

The fully specified family-face name of the font.

fontSize

The size in points to which the font is scaled.

Return Value
A font object for the specified name and size.

Discussion
The *fontName* is a fully specified family-face name, such as Helvetica-BoldOblique or Times-Roman. The *fontSize* is equivalent to using a font matrix of [*fontSize* 0 0 *fontSize* 0 0] with `fontWithDescriptor:size:` (page 1150). If you use a *fontSize* of 0.0, this method uses the default User Font size.

Fonts created with this method automatically flip themselves in flipped views. This method is the preferred means for creating fonts.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Cocoa OpenGL
CocoaVideoFrameToGWorld
From A View to A Movie
FunHouse
JSInterpreter

Declared In
NSFont.h

labelFontOfSize:

Returns the Aqua font used for standard interface labels in the specified size.

```
+ (NSFont *)labelFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize

The size in points to which the font is scaled.

Return Value
A font object of the specified size.

Discussion
If *fontSize* is 0 or negative, returns the label font with the default size.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CIVideoDemoGL
DockTile
TrackBall
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSFont.h

labelFontSize

Returns the size of the standard label font.

```
+ (CGFloat)labelFontSize
```

Return Value
The label font size in points.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSFont.h

menuBarFontOfSize:

Returns the font used for menu bar items, in the specified size.

```
+ (NSFont *)menuBarFontOfSize:(CGFloat)fontSize
```

Parameters
fontSize
The size in points to which the font is scaled.

Return Value
A font object of the specified size.

Discussion
If *fontSize* is 0 or negative, returns the menu bar font with the default size.

Availability
Available in Mac OS X v10.3 and later.

See Also
+ fontWithName:size: (page 1152)

Declared In
NSFont.h

menuFontOfSize:

Returns the font used for menu items, in the specified size.

```
+ (NSFont *)menuFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *fontSize* is 0 or negative, returns the menu items font with the default size.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ fontWithName:size:` (page 1152)

Related Sample Code

QTAudioContextInsert

QTAudioExtractionPanel

Declared In

`NSFont.h`

messageFontOfSize:

Returns the font used for standard interface items, such as button labels, menu items, and so on, in the specified size.

```
+ (NSFont *)messageFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *fontSize* is 0 or negative, returns this font at the default size. This method is equivalent to `systemFontOfSize:` (page 1157).

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ fontWithName:size:` (page 1152)

Related Sample Code

CIAnnotation

CIVideoDemoGL
ClockControl
MenuMadness

Declared In
NSFont.h

paletteFontOfSize:

Returns the font used for palette window title bars, in the specified size.

```
+ (NSFont *)paletteFontOfSize:(CGFloat)fontSize
```

Parameters
fontSize
The size in points to which the font is scaled.

Return Value
A font object of the specified size.

Discussion
If *fontSize* is 0 or negative, returns the palette title font at the default size.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ fontWithName:size: (page 1152)
+ titleBarFontOfSize: (page 1159)

Declared In
NSFont.h

preferredFontNames

Returns the names of fonts that the Application Kit tries first when a character has no font specified. (Deprecated in Mac OS X v10.4. The NSFontDescriptor constant NSFontCascadeListAttribute offers more powerful font substitution management.)

```
+ (NSArray *)preferredFontNames
```

Discussion
Returns the names of fonts that the Application Kit tries first when a character has no font specified or when the font specified doesn't have a glyph for that character. If none of these fonts provides a glyph, the remaining fonts on the system are searched for a glyph.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

See Also
+ setPreferredFontNames: (page 1156)

Declared In
NSFont.h

setPreferredFontNames:

Sets the list of preferred font names. (Deprecated in Mac OS X v10.4. The NSFontDescriptor constant NSFontCascadeListAttribute offers more powerful font substitution management.)

```
+ (void)setPreferredFontNames:(NSArray *)fontNames
```

Discussion
Sets the list of preferred font names to *fontNames* and records them in the user defaults database for all applications. The Application Kit tries these fonts first when a character has no font specified or when the font specified doesn't have a glyph for that character. If none of these fonts provides a glyph, the remaining fonts on the system are searched for a glyph.

This method is useful for optimizing glyph rendering for uncommon scripts, by guaranteeing that appropriate fonts are searched first. For example, suppose you have three hundred Latin alphabet fonts and one Cyrillic alphabet font. When you read a document in Russian, you want it to find the Cyrillic font quickly. Ordinarily, the Application Kit will search for the Cyrillic font among all 301 fonts. But if it is in the list of preferred fonts, the Cyrillic font will be one of the first searched.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

See Also
+ preferredFontNames (page 1155)

Declared In
NSFont.h

setUserFixedPitchFont:

Sets the font used by default for documents and other text under the user's control, when that font should be fixed-pitch, to the specified font.

```
+ (void)setUserFixedPitchFont:(NSFont *)aFont
```

Discussion
Specifying *aFont* as nil causes the default to be removed from the application domain.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ setUserFont: (page 1157)
+ userFixedPitchFontOfSize: (page 1161)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSFont.h

setUserFont:

Sets the font used by default for documents and other text under the user's control to the specified font.

```
+ (void)setUserFont:(NSFont *)aFont
```

Discussion
Specifying *aFont* as `nil` causes the default to be removed from the application domain.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ setUserFixedPitchFont:` (page 1156)
`+ userFontOfSize:` (page 1161)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSFont.h

smallSystemFontSize

Returns the size of the standard small system font.

```
+ (CGFloat)smallSystemFontSize
```

Return Value
The small system font size in points.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ButtonMadness
Quartz Composer QCTV
Sketch+Accessibility
SourceView
WhackedTV

Declared In
NSFont.h

systemFontOfSize:

Returns the Aqua system font used for standard interface items, such as button labels, menu items, and so on, in the specified size.

```
+ (NSFont *)systemFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize

The size in points to which the font is scaled.

Return Value
A font object of the specified size.

Discussion
If *fontSize* is 0 or negative, returns the system font at the default size.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ boldSystemFontOfSize: (page 1149)
+ userFontOfSize: (page 1161)
+ userFixedPitchFontOfSize: (page 1161)
+ fontWithName:size: (page 1152)

Related Sample Code
DatePicker
FunHouse
Quartz Composer QCTV
QuickLookDownloader
WhackedTV

Declared In
NSFont.h

systemFontSize

Returns the size of the standard system font.

```
+ (CGFloat)systemFontSize
```

Return Value
The standard system font size in points.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
iChatTheater
Mountains
PDFKitLinker2
PhotoSearch

Declared In
NSFont.h

systemFontSizeForControlSize:

Returns the font size used for the specified control size.

```
+ (CGFloat)systemFontSizeForControlSize:(NSControlSize)controlSize
```

Parameters

controlSize

The control size constant.

Return Value

The font size in points for the specified control size.

Discussion

If *controlSize* does not correspond to a valid `NSControlSize`, returns the size of the standard system font.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

IBFragmentView

ObjectPath

Declared In

`NSFont.h`

titleBarFontOfSize:

Returns the font used for window title bars, in the specified size.

```
+ (NSFont *)titleBarFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *fontSize* is 0 or negative, returns the title bar font at the default size. This method is equivalent to `boldSystemFontOfSize:` (page 1149).

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ paletteFontOfSize:` (page 1155)

Declared In

`NSFont.h`

toolTipsFontOfSize:

Returns the font used for tool tips labels, in the specified size.

```
+ (NSFont *)toolTipsFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize

The size in points to which the font is scaled.

Return Value

A font object of the specified size.

Discussion

If *fontSize* is 0 or negative, returns the tool tips font at the default size.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ fontWithName:size: (page 1152)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TipWrapper

Declared In

NSFont.h

useFont:

Records the given font name as one used in the current print operation. (Deprecated in Mac OS X v10.4. This is now automatically handled by Quartz.)

```
+ (void)useFont:(NSString *)fontName
```

Discussion

Records *fontName* as one used in the current print operation.

The NSFont class object keeps track of the fonts used in an NSView by recording each one that receives a set (page 1179) message. When the view is called upon to generate conforming PostScript language output (such as during printing), the NSFont class provides the list of fonts required for the %%DocumentFonts comment, as required by Adobe's document structuring conventions.

The useFont: argument augments this system by providing a way to register fonts that are included in the document but not set using NSFont's set (page 1179) method. For example, you might set a font by executing the setfont operator within a function created by the pswrap utility. In such a case, be sure to pair the use of the font with a useFont: message to register the font for listing in the document comments.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
NSFont.h

userFixedPitchFontOfSize:

Returns the font used by default for documents and other text under the user's control (that is, text whose font the user can normally change), when that font should be fixed-pitch, in the specified size.

```
+ (NSFont *)userFixedPitchFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize
The size in points to which the font is scaled.

Return Value
A font object of the specified size.

Discussion
If *fontSize* is 0 or negative, returns the fixed-pitch font at the default size.

The system does not guarantee that all the glyphs in a fixed-pitch font are the same width. For example, certain Japanese fonts are dual-pitch, and other fonts may have nonspacing marks that can affect the display of other glyphs.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ userFontOfSize: (page 1161)
+ fontWithName:size: (page 1152)
+ setUserFixedPitchFont: (page 1156)

Related Sample Code
ClipboardViewer
PTPPassThrough
Quartz Composer WWDC 2005 TextEdit
TextSizingExample

Declared In
NSFont.h

userFontOfSize:

Returns the font used by default for documents and other text under the user's control (that is, text whose font the user can normally change), in the specified size.

```
+ (NSFont *)userFontOfSize:(CGFloat)fontSize
```

Parameters

fontSize
The size in points to which the font is scaled.

Return Value
A font object of the specified size.

Discussion
If *fontSize* is 0 or negative, returns the user font at the default size.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ userFixedPitchFontOfSize: (page 1161)
+ fontWithName:size: (page 1152)
+ setUserFont: (page 1157)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
UIElementInspector

Declared In
NSFont.h

Instance Methods

advancementForGlyph:

Returns the nominal spacing for the given glyph—the distance the current point moves after showing the glyph—accounting for the receiver's size.

```
- (NSSize)advancementForGlyph:(NSGlyph)aGlyph
```

Parameters

aGlyph
The glyph whose advancement is returned.

Return Value
The advancement spacing in points.

Discussion
This spacing is given according to the glyph's movement direction, which is either strictly horizontal or strictly vertical.

Availability
Available in Mac OS X v10.0 and later.

See Also
- boundingRectForGlyph: (page 1164)
- maximumAdvancement (page 1172)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSFont.h

afmDictionary

Returns the AFM font's dictionary. (Deprecated in Mac OS X v10.4. Use accessor functions listed in "Keys to the AFM Dictionary" (page 1184) instead.)

```
- (NSDictionary *)afmDictionary
```

Discussion
Always returns nil.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
NSFont.h

ascender

Returns the top y-coordinate, offset from the baseline, of the receiver's longest ascender.

```
- (CGFloat)ascender
```

Return Value
The distance of the longest ascender's top y-coordinate from the baseline in points.

Availability
Available in Mac OS X v10.0 and later.

See Also
- descender (page 1166)
- capHeight (page 1164)
- xHeight (page 1182)

Related Sample Code
GeekGameBoard
NSFontAttributeExplorer
QTKitTimeCode

Declared In
NSFont.h

boundingRectForFont

Returns the receiver's bounding rectangle, scaled to the font's size.

```
- (NSRect)boundingRectForFont
```

Discussion
The bounding rectangle is the union of the bounding rectangles of every glyph in the font.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `boundingRectForGlyph:` (page 1164)

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

boundingRectForGlyph:

Returns the bounding rectangle for the specified glyph, scaled to the receiver's size.

```
- (NSRect)boundingRectForGlyph:(NSGlyph)aGlyph
```

Discussion
Japanese fonts encoded with the scheme "EUC12-NJE-CFEncoding" do not have individual metrics or bounding boxes available for the glyphs above 127. For those glyphs, this method returns the bounding rectangle for the font instead.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `boundingRectForFont` (page 1163)
- `getBoundingRects:forGlyphs:count:` (page 1169)
- `getAdvancements:forGlyphs:count:` (page 1168)
- `getAdvancements:forPackedGlyphs:length:` (page 1168)

Declared In
`NSFont.h`

capHeight

Returns the receiver's cap height.

```
- (CGFloat)capHeight
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `ascender` (page 1163)
- `descender` (page 1166)
- `xHeight` (page 1182)

Related Sample Code
NSFontAttributeExplorer

Declared In
NSFont.h

coveredCharacterSet

Returns an NSCharacterSet object containing all of the nominal characters renderable by the receiver, which is all of the entries mapped in the receiver's 'cmap' table.

```
- (NSCharacterSet *)coveredCharacterSet
```

Return Value
An NSCharacterSet object containing all of the nominal characters renderable by the receiver.

Discussion
The number of glyphs supported by a given font is often larger than the number of characters contained in the character set returned by this method.

Availability
Available in Mac OS X v10.2 and later.

Declared In
NSFont.h

defaultLineHeightForFont

Returns the default line height for the receiver. (Deprecated in Mac OS X v10.4. Use the NSLayoutManager method defaultLineHeightForFont: (page 1456) instead.)

```
- (CGFloat)defaultLineHeightForFont
```

Discussion
Equivalent to ascent plus descent plus linegap.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

See Also
- ascender (page 1163)
- descender (page 1166)

Related Sample Code
NSFontAttributeExplorer

Declared In
NSFont.h

descender

Returns the bottom y coordinate, offset from the baseline, of the receiver's longest descender.

```
- (CGFloat)descender
```

Discussion
Thus, if the longest descender extends 2 points below the baseline, `descender` will return –2.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DockTile
GeekGameBoard
NSFontAttributeExplorer
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSFont.h`

displayName

Returns the name, including family and face, used to represent the font in the user interface, typically localized for the user's language.

```
- (NSString *)displayName
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
NSFontAttributeExplorer
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSFont.h`

encodingScheme

Returns the name of the receiver's encoding scheme. (Deprecated in Mac OS X v10.4. Use `mostCompatibleStringEncoding` (page 1173) instead.)

```
- (NSString *)encodingScheme
```

Discussion
Returns the name of the receiver's encoding scheme, such as "AdobeStandardEncoding," "ISOLatin1Encoding," "FontSpecific," and so on.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
NSFont.h

familyName

Returns the receiver's family name—for example, "Times" or "Helvetica."

```
- (NSString *)familyName
```

Discussion
This name is the one that `NSFontManager` uses and may differ slightly from the AFM name.

The value returned by this method is intended for an application's internal usage and not for display. Use `displayName` (page 1166) instead.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- fontName` (page 1167)

Declared In
NSFont.h

fontDescriptor

Returns the receiver's font descriptor.

```
- (NSFontDescriptor *)fontDescriptor
```

Return Value
A font descriptor object that describes the receiver.

Discussion
The font descriptor contains a mutable dictionary of optional attributes for creating an `NSFont` object. See documentation on NSFontDescriptor for more information.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
CoreTextArcCocoa

Declared In
NSFont.h

fontName

Returns the receiver's full font name, as used in PostScript language code—for example, "Times-Roman" or "Helvetica-Oblique."

```
- (NSString *)fontName
```

Discussion
The value returned by this method is intended for an application's internal usage and not for display. Use `displayName` (page 1166) instead.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `familyName` (page 1167)

Related Sample Code
FunHouse

Declared In
`NSFont.h`

getAdvancements:forGlyphs:count:

Returns an array of the advancements for the specified glyphs rendered by the receiver.

```
- (void)getAdvancements:(NSSizeArray)advancements forGlyphs:(const NSGlyph *)glyphs
    count:(NSUInteger)glyphCount
```

Discussion
Returns in *advancements* an array of the advancements for the glyphs specified by *glyphs* and rendered by the receiver. The *glyphCount* must specify the count of glyphs passed in *glyphs*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `boundingRectForFont` (page 1163)
- `boundingRectForGlyph:` (page 1164)
- `getAdvancements:forPackedGlyphs:length:` (page 1168)
- `getBoundingRects:forGlyphs:count:` (page 1169)

Declared In
`NSFont.h`

getAdvancements:forPackedGlyphs:length:

Returns an array of the advancements for the specified packed glyphs and rendered by the receiver.

```
- (void)getAdvancements:(NSSizeArray)advancements forPackedGlyphs:(const void
  *)packedGlyphs length:(NSUInteger)length- (void)getAdvancements
```

Discussion
Returns in *advancements* an array of the advancements for the packed glyphs specified by *packedGlyphs* and rendered by the receiver. The *glyphCount* must specify the count of glyphs passed in *packedGlyphs*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `boundingRectForFont` (page 1163)
- `boundingRectForGlyph:` (page 1164)
- `getBoundingRects:forGlyphs:count:` (page 1169)
- `getAdvancements:forGlyphs:count:` (page 1168)

Declared In
`NSFont.h`

getBoundingRects:forGlyphs:count:

Returns an array of the bounding rectangles for the specified glyphs rendered by the receiver.

```
- (void)getBoundingRects:(NSRectArray)bounds forGlyphs:(const NSGlyph *)glyphs
    count:(NSUInteger)glyphCount
```

Discussion
Returns in *bounds* an array of the bounding rectangles for the glyphs specified by *glyphs* and rendered by the receiver. The *glyphCount* must specify the count of glyphs passed in *glyphs*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `boundingRectForFont` (page 1163)
- `boundingRectForGlyph:` (page 1164)
- `getAdvancements:forGlyphs:count:` (page 1168)
- `getAdvancements:forPackedGlyphs:length:` (page 1168)

Declared In
`NSFont.h`

glyphIsEncoded:

Returns a Boolean value indicating whether the receiver encodes the given glyph. (Deprecated in Mac OS X v10.4. The value can be deduced by `aGlyph < [NSFont numberOfGlyphs]` since only `NSNativeShortGlyphPacking` is supported.)

```
- (BOOL)glyphIsEncoded:(NSGlyph)aGlyph
```

Discussion
Returns `YES` if the receiver encodes *aGlyph*, `NO` if it doesn't contain it.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
NSFont.h

glyphPacking

Returns the best way to encode the receiver's glyphs into an array of bytes. (Deprecated in Mac OS X v10.4. Only `NSNativeShortGlyphPacking` (page 1184) is supported.)

```
- (NSMultibyteGlyphPacking)glyphPacking
```

Discussion
Returns the best way to encode the receiver's glyphs into an array of bytes. The return value is one of values described in "Constants" (page 1182).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
NSFont.h

glyphWithName:

Returns the named encoded glyph, or –1 if the receiver contains no such glyph.

```
- (NSGlyph)glyphWithName:(NSString *)glyphName
```

Parameters
glyphName
The name of the glyph.

Return Value
The named encoded glyph.

Discussion
Returns –1 if the glyph named *glyphName* isn't encoded.

Glyph names in fonts do not always accurately identify the glyph. If possible, look up the appropriate glyph on your own.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSFont.h

isBaseFont

Returns a Boolean value indicating whether the receiver is a PostScript base font. (Deprecated in Mac OS X v10.4. This information is not relevant to Mac OS X.)

```
- (BOOL)isBaseFont
```

Discussion
Returns `YES` if the receiver is a PostScript base font, `NO` if it's a PostScript composite font composed of other base fonts.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`NSFont.h`

isFixedPitch

Returns a Boolean value indicating whether all glyphs in the receiver have the same advancement.

```
- (BOOL)isFixedPitch
```

Return Value
`YES` if all glyphs in the receiver have the same advancement; `NO` if any advancements differ.

Discussion
Some Japanese fonts encoded with the scheme "EUC12-NJE-CFEncoding" return that they have the same advancement, but actually encode glyphs with one of two advancements, for historical compatibility. You may need to handle such fonts specially for some applications.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- advancementForGlyph:` (page 1162)

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

italicAngle

Returns the receiver's italic angle, the amount that the font is slanted in degrees counterclockwise from the vertical, as read from its AFM file. Because the slant is measured counterclockwise, English italic fonts typically return a negative value.

```
- (CGFloat)italicAngle
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
NSFontAttributeExplorer

Declared In
NSFont.h

leading

Returns the receiver's leading.

```
- (CGFloat)leading
```

Availability
Available in Mac OS X v10.4 and later.

See Also
+ fontWithDescriptor:size: (page 1150)

Related Sample Code
NSFontAttributeExplorer

Declared In
NSFont.h

matrix

Returns the receiver's font matrix, a standard six-element transformation matrix as used in the PostScript language, specifically with the `makefont` operator.

```
- (const CGFloat *)matrix
```

Discussion
In most cases, with a font of `fontSize`, this matrix is [`fontSize` 0 0 `fontSize` 0 0].

Availability
Available in Mac OS X v10.0 and later.

See Also
+ fontWithDescriptor:size: (page 1150)

Declared In
NSFont.h

maximumAdvancement

Returns the greatest advancement of any of the receiver's glyphs.

```
- (NSSize)maximumAdvancement
```

Discussion
This advancement is always either strictly horizontal or strictly vertical.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `advancementForGlyph:` (page 1162)

Related Sample Code
NSFontAttributeExplorer
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSFont.h`

mostCompatibleStringEncoding

Returns the string encoding that works best with the receiver, where there are the fewest possible unmatched characters in the string encoding and glyphs in the font.

```
- (NSStringEncoding)mostCompatibleStringEncoding
```

Return Value
The string encoding that works best with the receiver.

Discussion
You can use `NSString`'s `dataUsingEncoding:` or `dataUsingEncoding:allowLossyConversion:` method to convert the string to this encoding.

If this method returns `NSASCIIStringEncoding`, it could not determine the correct encoding and assumed that the font can render only ASCII characters.

This method works heuristically using well-known font encodings, so for nonstandard encodings it may not in fact return the optimal string encoding.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

numberOfGlyphs

Returns the number of glyphs in the receiver.

```
- (NSUInteger)numberOfGlyphs
```

Discussion
Glyphs are numbered starting at 0.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
NSFontAttributeExplorer

Declared In
NSFont.h

pointSize

Returns the receiver's point size, or the effective vertical point size for a font with a nonstandard matrix.

```
- (CGFloat)pointSize
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
FunHouse
GeekGameBoard
Quartz Composer WWDC 2005 TextEdit

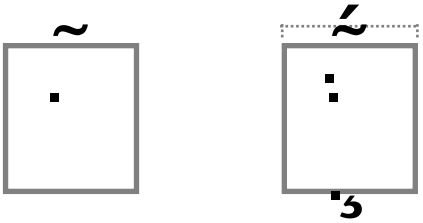
Declared In
NSFont.h

positionOfGlyph:forCharacter:struckOverRect:

Calculates and returns a suitable location for the given glyph to be drawn. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSPoint)positionOfGlyph:(NSGlyph)aGlyph forCharacter:(unichar)aChar
    struckOverRect:(NSRect)aRect
```

Discussion
Calculates and returns a suitable location for *aGlyph* to be drawn as a diacritic or nonspacing mark relative to *aRect*, assuming that *aGlyph* represents *aChar*. Returns NSZeroPoint if the location can't be calculated. The nature of *aChar* as one appearing above or below its base character determines the location returned. For example, in the first figure below, the gray tilde and box represent *aGlyph* and *aRect*, and the black dot is the point returned (defined relative to the origin of the *aRect*).



To place multiple glyphs with respect to a rectangle, work from the innermost glyphs to the outermost. As you calculate the position of each glyph, enlarge the rectangle to include the bounding rectangle of the glyph in preparation for the next glyph. The second figure shows a tilde, acute accent, and cedilla all placed in their appropriate positions with respect to a rectangle, with the acute accent placed relative to the expanded bounding box of the base rectangle and the tilde.

This method is the last fallback mechanism for performing minimally legible typography when metrics aren't available. Use it when `positionOfGlyph:struckOverGlyph:metricsExist:` (page 1175) indicates that metrics don't exist for the base glyph specified, or when you are combining glyphs from different fonts (for example, the base glyph is in a different font than the accent). It can account for the layout and placement of most Latin, Greek, and Cyrillic nonspacing marks. You should draw the glyph at the returned location, even if it's `NSZeroRect`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`NSFont.h`

positionOfGlyph:precededByGlyph:isNominal:

Calculates and returns the location of a glyph. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSPoint)positionOfGlyph:(NSGlyph)aGlyph precededByGlyph:(NSGlyph)prevGlyph
    isNominal:(BOOL *)flag
```

Discussion
Calculates and returns the location of *aGlyph* relative to *prevGlyph*, assuming that *prevGlyph* precedes it in the layout (not necessarily in the character stream). The point returned should be used relative to whatever location is used for *prevGlyph*. If *flag* is non-`nil`, it's filled with `NO` if kerning tables are available and were used in the calculation; it is filled with `YES` if the default spacing is used.

Returns `NSZeroPoint` if either *aGlyph* or *prevGlyph* is `NSControlGlyph` or is invalid. Returns the nominal advancement of *prevGlyph* if *aGlyph* is `NSNullGlyph`.

This method is useful for sequential glyph placement when glyphs aren't drawn with a single PostScript operation.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`NSFont.h`

positionOfGlyph:struckOverGlyph:metricsExist:

Calculates and returns a suitable location for the given glyph to be drawn. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSPoint)positionOfGlyph:(NSGlyph)aGlyph struckOverGlyph:(NSGlyph)baseGlyph
    metricsExist:(BOOL *)flag
```

Discussion
Calculates and returns a suitable location for *aGlyph* to be drawn as a diacritic or nonspacing mark relative to *baseGlyph*. The point returned should be used relative to whatever location is used for *baseGlyph*. If *flag* is non-nil it's filled with YES if font metrics are available, NO if they're not. If flag is returned as NO, the result isn't valid and shouldn't be used. In that case, use
positionOfGlyph:struckOverRect:metricsExist: (page 1176) or
positionOfGlyph:forCharacter:struckOverRect: (page 1174) to calculate a reasonable offset.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

See Also
- positionsForCompositeSequence:numberOfGlyphs:pointArray: (page 1177)
- positionOfGlyph:struckOverRect:metricsExist: (page 1176)

Declared In
NSFont.h

positionOfGlyph:struckOverRect:metricsExist:

Overridden by subclasses to calculate and return a suitable location for a glyph to be drawn. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSPoint)positionOfGlyph:(NSGlyph)aGlyph struckOverRect:(NSRect)aRect
    metricsExist:(BOOL *)flag
```

Discussion
Overridden by subclasses to calculate and return a suitable location for *aGlyph* to be drawn as a diacritic or nonspacing mark relative to *aRect*, provided metrics exist. Returns NSZeroRect if the location can't be determined. If *flag* is non-nil it's filled with YES if font metrics are available, NO if they're not. If *flag* is returned as NO, the result isn't valid and shouldn't be used. In that case, use
positionOfGlyph:forCharacter:struckOverRect: (page 1174) to calculate a reasonable offset.

Because current PostScript font metrics don't include support for generic placement relative to rectangles, NSFont's implementation of this method always returns NSZeroPoint and returns *flag* as NO.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
NSFont.h

positionOfGlyph:withRelation:toBaseGlyph:totalAdvancement:metricsExist:

Calculates and returns a suitable location for a glyph to be drawn. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSPoint)positionOfGlyph:(NSGlyph)aGlyph withRelation:(NSGlyphRelation)relation
    toBaseGlyph:(NSGlyph)baseGlyph totalAdvancement:(NSSizePointer)offset
  metricsExist:(BOOL *)flag
```

Discussion
Calculates and returns a suitable location for *aGlyph* to be drawn relative to *baseGlyph*, where *relation* is `NSGlyphBelow` or `NSGlyphAbove`. The point returned should be used relative to whatever location is used for *baseGlyph*. This method is useful for calculating the layout of stacked glyphs, found in some non-Western scripts.

If *offset* is non-`NULL`, this method sets it to the larger of the two glyphs' advancements, allowing for reasonable layout of following glyphs.

If *flag* is non-`nil`, this method sets it to whether font metrics are available: `YES` if they are, `NO` if they're not. If metrics aren't available, the location is calculated as a simple stacking with no gap between *baseGlyph* and *aGlyph*. Current Postscript fonts do not contain appropriate font metrics, so this method always sets *flag* to `NO`. If you subclass `NSFont` to handle fonts that do contain metrics, override this method.

This method supports only horizontally laid out base glyphs.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`NSFont.h`

positionsForCompositeSequence:numberOfGlyphs:pointArray:

Calculates glyph locations. (Deprecated in Mac OS X v10.4. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

```
- (NSInteger)positionsForCompositeSequence:(NSGlyph *)glyphs
    numberOfGlyphs:(NSInteger)numGlyphs pointArray:(NSPointArray)points
```

Discussion
Calculates and fills *points* with the locations for *glyphs*, assuming the first glyph is a base character and those following are nonspacing marks. These points should all be interpreted as relative to the location of the first glyph in *glyphs*. The storage block *points* points to should be large enough for at least *numGlyphs* points. Returns the number of points that could be calculated.

If the number of points calculated is less than *numGlyphs*, the number of glyphs provided, you can use `positionOfGlyph:struckOverRect:metricsExist:` (page 1176) to determine the positions for the remaining glyphs. When using that method, calculate the base rectangle for each glyph from the bounding rectangles and positions of all preceding glyphs.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`NSFont.h`

printerFont

Returns the scalable PostScript font corresponding to itself.

```
- (NSFont *)printerFont
```

Discussion
When sent to a font object representing a scalable PostScript font, returns `self`. When sent to a font object representing a bitmapped screen font, returns its corresponding scalable PostScript font.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `screenFont` (page 1178)

Related Sample Code
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSFont.h`

renderingMode

Returns the rendering mode of the receiver.

```
- (NSFontRenderingMode)renderingMode
```

Return Value
The rendering mode of the receiver.

Discussion
For valid rendering modes, see "Constants" (page 1182).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `screenFontWithRenderingMode:` (page 1179)

Declared In
`NSFont.h`

screenFont

Returns the bitmapped screen font corresponding to itself.

```
- (NSFont *)screenFont
```

Discussion
When sent to a font object representing a scalable PostScript font, returns a bitmapped screen font matching the receiver in typeface and matrix (or size), or `nil` if such a font can't be found. When sent to a font object representing a bitmapped screen font, returns `nil`.

Screen fonts are for direct use with the window server only. Never use them with Application Kit objects, such as in `setFont:` methods. Internally, the Application Kit automatically uses the corresponding screen font for a font object as long as the view is not rotated or scaled.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `printerFont` (page 1178)
- `screenFontWithRenderingMode:` (page 1179)

Declared In
`NSFont.h`

screenFontWithRenderingMode:

Returns a bitmapped screen font, when sent to a font object representing a scalable PostScript font, with the specified rendering mode, matching the receiver in typeface and matrix (or size), or `nil` if such a font can't be found.

```
- (NSFont *)screenFontWithRenderingMode:(NSFontRenderingMode)renderingMode
```

Discussion
For valid rendering modes, see `NSFontRenderingMode` (page 1182).

Screen fonts are for direct use with the window server only. Never use them with Application Kit objects, such as in `setFont:` methods. Internally, the Application Kit automatically uses the corresponding screen font for a font object as long as the view is not rotated or scaled.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `printerFont` (page 1178)
- `screenFont` (page 1178)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSFont.h`

set

Sets this font as the font for the current graphics context.

```
- (void)set
```

Discussion
This method sets the font for the graphics system but does not affect the higher-level settings of the Cocoa text system, which are controlled by text attributes.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ useFont: (page 1160)
- setInContext: (page 1180)

Declared In
NSFont.h

setInContext:

Sets this font as the font for the specified graphics context.

```
- (void)setInContext:(NSGraphicsContext *)graphicsContext
```

Parameters
graphicsContext
The graphics context for which the font is set.

Discussion
This method sets the font for the graphics system but does not affect the higher-level settings of the Cocoa text system, which are controlled by text attributes.

Availability
Available in Mac OS X v10.4 and later.

See Also
+ useFont: (page 1160)
- set (page 1179)

Declared In
NSFont.h

textTransform

Returns the current transformation matrix for the receiver.

```
- (NSAffineTransform *)textTransform
```

Availability
Available in Mac OS X v10.4 and later.

See Also
+ useFont: (page 1160)
- set (page 1179)

Declared In
NSFont.h

underlinePosition

Returns the baseline offset that should be used when drawing underlines with the receiver, as determined by the font's AFM file.

```
- (CGFloat)underlinePosition
```

Discussion
This value is usually negative, which must be considered when drawing in a flipped coordinate system.

Availability
Available in Mac OS X v10.0 and later.

See Also
- underlineThickness (page 1181)

Related Sample Code
NSFontAttributeExplorer

Declared In
NSFont.h

underlineThickness

Returns the thickness that should be used when drawing underlines with the receiver, as determined by the font's AFM file.

```
- (CGFloat)underlineThickness
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- underlinePosition (page 1181)

Related Sample Code
NSFontAttributeExplorer

Declared In
NSFont.h

widthOfString:

Returns the x-axis offset of the current point when the specified string is drawn with a `show` operator in the receiving font. (Deprecated in Mac OS X v10.4. Use the Application Kit string-drawing methods, as described in NSString Additions (page 2581).)

```
- (CGFloat)widthOfString:(NSString *)aString
```

Discussion
This method is for backward compatibility only. This method performs lossy conversion of *aString* to the most compatible encoding for the receiving font. Use this method only when you're sure all of *aString* can be rendered with the receiving font.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

See Also
- `mostCompatibleStringEncoding` (page 1173)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSFont.h`

xHeight

Returns the x-height of the receiver.

```
- (CGFloat)xHeight
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `ascender` (page 1163)
- `descender` (page 1166)

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFont.h`

Constants

NSFontRenderingMode

These constants specify the font rendering mode.

```
typedef enum {
    NSFontDefaultRenderingMode = 0,
    NSFontAntialiasedRenderingMode = 1,
    NSFontIntegerAdvancementsRenderingMode = 2,
    NSFontAntialiasedIntegerAdvancementsRenderingMode = 3
} NSFontRenderingMode;
```

Constants

`NSFontDefaultRenderingMode`

Determines the actual mode based on the user preference settings.

Available in Mac OS X v10.4 and later.

Declared in `NSFont.h`.

`NSFontAntialiasedRenderingMode`

Specifies antialiased, floating-point advancements rendering mode (synonymous with printerFont).

Available in Mac OS X v10.4 and later.

Declared in `NSFont.h`.

`NSFontIntegerAdvancementsRenderingMode`

Specifies integer advancements rendering mode.

Available in Mac OS X v10.4 and later.

Declared in `NSFont.h`.

`NSFontAntialiasedIntegerAdvancementsRenderingMode`

Specifies antialiased, integer advancements rendering mode.

Available in Mac OS X v10.4 and later.

Declared in `NSFont.h`.

Declared In

`NSFont.h`

PostScript Transformation Matrix

The identity matrix.

```
const float *NSFontIdentityMatrix;
```

Constants

`NSFontIdentityMatrix`

A transformation matrix useful as a parameter to fontWithDescriptor:size: (page 1150).

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

Declared In

`NSFont.h`

NSMultibyteGlyphPacking

A constant for glyph packing.

```
typedef enum {
     NSNativeShortGlyphPacking = 5
} NSMultibyteGlyphPacking;
```

Constants

`NSNativeShortGlyphPacking`

The native format for Mac OS X.

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

Discussion

Cocoa stores all text data as Unicode. The text system converts Unicode into glyph IDs and places them in 1-, 2-, or 4-byte storage depending on the context. To render text, you must convert the storage into a format the text engine understands. The following constants describe the glyph packing schemes the text rendering engine can use. They are used to extract glyphs from a font for making a multibyte (or single-byte) array of glyphs for passing to an interpreter, such as the window server, which expects a big-endian multibyte stream (that is, "packed glyphs") instead of a pure `NSGlyph` stream. They're used by `glyphPacking` (page 1170). With Quartz, the engine always expects the format to be in 2-byte short array, so `NSNativeShortGlyphPacking` is the only format currently in use.

Declared In

`NSFont.h`

Reserved Glyph Codes

These constants define reserved glyph codes.

```
enum {
   NSControlGlyph = 0x00FFFFFF,
   NSNullGlyph = 0x0
};
```

Constants

`NSControlGlyph`

`NSGlyphGenerator` generates `NSControlGlyph` for all characters in the Unicode General Category C* and U200B (ZERO WIDTH SPACE).

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

`NSNullGlyph`

A null glyph.

Available in Mac OS X v10.0 and later.

Declared in `NSFont.h`.

Declared In

`NSFont.h`

Keys to the AFM Dictionary

These constants are used as keys retrieve information from an AFM dictionary. (Deprecated. The AFM dictionary is no longer used in Mac OS X. Use the font metrics accessor methods listed with the individual constants instead.)

```
NSString *NSAFMFamilyName;
NSString *NSAFMFontName;
NSString *NSAFMFormatVersion;
NSString *NSAFMFullName;
NSString *NSAFMNotice;
NSString *NSAFMVersion;
NSString *NSAFMWeight;
NSString *NSAFMEncodingScheme;
NSString *NSAFMCharacterSet;
NSString *NSAFMCapHeight;
NSString *NSAFMXHeight;
NSString *NSAFMAscender;
NSString *NSAFMDescender;
NSString *NSAFMUnderlinePosition;
NSString *NSAFMUnderlineThickness;
NSString *NSAFMItalicAngle;
NSString *NSAFMMappingScheme;
```

Constants

`NSAFMFamilyName`

Font family name key. (Deprecated. Use `familyName` (page 1167) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMFontName`

Font name key. (Deprecated. Use `displayName` (page 1166) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMFormatVersion`

Format version name key. (Deprecated. This information is not relevant to Mac OS X.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMFullName`

Full font name key. (Deprecated. Use `fontName` (page 1167) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMNotice`

Font notice key. (Deprecated. Use Apple Type Services instead.)

Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.
Declared in `NSFont.h`.

`NSAFMVersion`

Font version key. (Deprecated. Use Apple Type Services instead.)

Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.
Declared in `NSFont.h`.

`NSAFMWeight`

Font weight key. (Deprecated. Use the `NSFontManager` method `weightOfFont:` (page 1239) instead.)

Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.
Declared in `NSFont.h`.

`NSAFMEncodingScheme`

Font encoding scheme key. (Deprecated. Use `mostCompatibleStringEncoding` (page 1173) instead.)

Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.
Declared in `NSFont.h`.

`NSAFMCharacterSet`

Font character set key. (Deprecated. Use `coveredCharacterSet` (page 1165) instead.)

Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.
Declared in `NSFont.h`.

`NSAFMCapHeight`

Font cap-height key. (Deprecated. Use `capHeight` (page 1164) instead.)

Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.
Declared in `NSFont.h`.

`NSAFMXHeight`

Font x-height key. (Deprecated. Use `xHeight` (page 1182) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMAscender`

Font ascender height key. (Deprecated. Use `ascender` (page 1163) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMDescender`

Font descender depth key. (Deprecated. Use `descender` (page 1166) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMUnderlinePosition`

Font underline rule position key. (Deprecated. Use `underlinePosition` (page 1181) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMUnderlineThickness`

Font underline rule thickness key. (Deprecated. Use `underlineThickness` (page 1181) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMItalicAngle`

Font italic angle key. (Deprecated. Use `italicAngle` (page 1171) instead.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

`NSAFMMappingScheme`

Font mapping scheme key. (Deprecated. This information is irrelevant to Mac OS X.)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared in `NSFont.h`.

Declared In
`NSFont.h`

NSGlyph

This type is used to specify glyphs in such methods as `glyphWithName:`.

```
typedef unsigned int NSGlyph;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFont.h`

NSGlyphRelation

These constants are used for calculating the layout of stacked glyphs. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage)

```
typedef enum _NSGlyphRelation {
    NSGlyphBelow = 1,
    NSGlyphAbove = 2
} NSGlyphRelation;
```

Constants
`NSGlyphBelow`

The glyph is located below the base glyph. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in `NSFont.h`.

`NSGlyphAbove`

The glyph is located above the base glyph. (Deprecated. Context-sensitive interglyph spacing is now performed at the typesetting stage.)

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in `NSFont.h`.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
NSFont.h

NSMultibyteGlyphPacking

Glyph packing constants are used to extract glyphs from a font for making a multibyte (or single-byte) array of glyphs for passing to an interpreter, such as the window server. With Quartz, the engine always expects the format to be in 2-byte short array, so NSNativeShortGlyphPacking (page 1184) is the only format currently in use. (Deprecated. Use NSNativeShortGlyphPacking (page 1184) instead.)

```
enum {
    NSOneByteGlyphPacking,
    NSJapaneseEUCGlyphPacking,
    NSAsciiWithDoubleByteEUCGlyphPacking,
    NSTwoByteGlyphPacking,
    NSFourByteGlyphPacking,
}
```

Constants

NSOneByteGlyphPacking

One-byte storage format. (Deprecated. Use NSNativeShortGlyphPacking (page 1184) instead.)

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in NSFont.h.

NSJapaneseEUCGlyphPacking

Extended Unix Code for Japanese format. (Deprecated. Use NSNativeShortGlyphPacking (page 1184) instead.)

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in NSFont.h.

NSAsciiWithDoubleByteEUCGlyphPacking

Two-byte Extended Unix Code format. (Deprecated. Use NSNativeShortGlyphPacking (page 1184) instead.)

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in NSFont.h.

NSTwoByteGlyphPacking

Two-byte storage format. (Deprecated. Use NSNativeShortGlyphPacking (page 1184) instead.)

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in NSFont.h.

NSFourByteGlyphPacking

Four-byte storage format. (Deprecated. Use NSNativeShortGlyphPacking (page 1184) instead.)

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in NSFont.h.

Declared In
NSFont.h

Notifications

NSAntialiasThresholdChangedNotification

Posted after the threshold for anti-aliasing changes.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSFont.h`

NSFontSetChangedNotification

Posted after the the currently-set font changes.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSFont.h`

NSFontDescriptor Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSFontDescriptor.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Font Handling

Overview

`NSFontDescriptor` objects provide a mechanism to describe a font with a dictionary of attributes. This font descriptor can be used later to create or modify an `NSFont` object. Mac OS X v10.4 and later provides a font matching capability, so that you can partially describe a font by creating a font descriptor with, for example, just a family name. You can then find all the available fonts on the system with a matching family name using `matchingFontDescriptorsWithMandatoryKeys:` (page 1198).

There are several ways to create a new `NSFontDescriptor` object. You can use `alloc` and `initWithFontAttributes:` (page 1197), `fontDescriptorWithFontAttributes:` (page 1193), `fontDescriptorWithName:matrix:` (page 1193), or `fontDescriptorWithName:size:` (page 1194). to create a font descriptor based on either your custom attributes dictionary or on a specific font's name and size. Alternatively you can use one of the `fontDescriptor...` instance methods (such as `fontDescriptorWithFace:` (page 1195)) to create a modified version of an existing descriptor. The latter methods are useful if you have an existing descriptor and simply want to change one aspect.

All attributes in the attributes dictionary are optional.

Adopted Protocols

NSCoding

- `encodeWithCoder:`
- `initWithCoder:`

NSCopying

- `copyWithZone:`

Tasks

Creating a Font Descriptor

`+ fontDescriptorWithFontAttributes:` (page 1193)
Returns a font descriptor with a dictionary of attributes.

`+ fontDescriptorWithName:matrix:` (page 1193)
Returns a font descriptor with the `NSFontNameAttribute` and `NSFontMatrixAttribute` dictionary attributes set to the given values.

`+ fontDescriptorWithName:size:` (page 1194)
Returns a font descriptor with the `NSFontNameAttribute` and `NSFontSizeAttribute` dictionary attributes set to the given values.

`- fontDescriptorByAddingAttributes:` (page 1195)
Returns a new font descriptor that is the same as the receiver but with the specified attributes taking precedence over the existing ones.

`- fontDescriptorWithFace:` (page 1195)
Returns a new font descriptor that is the same as the receiver but with the specified face.

`- fontDescriptorWithFamily:` (page 1196)
Returns a new font descriptor whose attributes are the same as the receiver but from the specified family.

`- fontDescriptorWithMatrix:` (page 1196)
Returns a new font descriptor that is the same as the receiver but with the specified matrix.

`- fontDescriptorWithSize:` (page 1196)
Returns a new font descriptor that is the same as the receiver but with the specified point size.

`- fontDescriptorWithSymbolicTraits:` (page 1197)
Returns a new font descriptor that is the same as the receiver but with the specified symbolic traits taking precedence over the existing ones.

Initializing a Font Descriptor

`- initWithFontAttributes:` (page 1197)
Initializes and returns a new font descriptor with the specified attributes.

Finding Fonts

`- matchingFontDescriptorsWithMandatoryKeys:` (page 1198)
Returns all the fonts available on the system whose specified attributes match those of the receiver.

`- matchingFontDescriptorWithMandatoryKeys:` (page 1198)
Returns a normalized font descriptor whose specified attributes match those of the receiver.

Querying a Font Descriptor

- `fontAttributes` (page 1194)
  Returns the receiver's dictionary of attributes.
- `matrix` (page 1199)
  Returns the current transform matrix of the receiver.
- `objectForKey:` (page 1199)
  Returns the font attribute specified by the given key.
- `pointSize` (page 1199)
  Returns the point size of the receiver.
- `postscriptName` (page 1200)
  Returns the PostScript name of the receiver.
- `symbolicTraits` (page 1200)
  Returns a bit mask that describes the traits of the receiver.

Class Methods

fontDescriptorWithFontAttributes:

Returns a font descriptor with a dictionary of attributes.

```
+ (NSFontDescriptor *)fontDescriptorWithFontAttributes:(NSDictionary *)attributes
```

Parameters

attributes
  The attributes for the font descriptor. If `nil`, the font descriptor's dictionary will be empty.

Return Value
The new font descriptor.

Availability
Available in Mac OS X v10.3 and later.

See Also
`+ fontDescriptorWithName:matrix:` (page 1193)
`+ fontDescriptorWithName:matrix:` (page 1193)

Declared In
`NSFontDescriptor.h`

fontDescriptorWithName:matrix:

Returns a font descriptor with the `NSFontNameAttribute` and `NSFontMatrixAttribute` dictionary attributes set to the given values.

```
+ (NSFontDescriptor *)fontDescriptorWithName:(NSString *)fontName
    matrix:(NSAffineTransform *)matrix
```

Parameters

fontName

The value for `NSFontNameAttribute`.

matrix

The value for `NSFontMatrixAttribute`.

Return Value
The new font descriptor.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ fontDescriptorWithFontAttributes:` (page 1193)
`+ fontDescriptorWithName:size:` (page 1194)

Declared In
`NSFontDescriptor.h`

fontDescriptorWithName:size:

Returns a font descriptor with the `NSFontNameAttribute` and `NSFontSizeAttribute` dictionary attributes set to the given values.

```
+ (NSFontDescriptor *)fontDescriptorWithName:(NSString *)fontName size:(CGFloat)size
```

Parameters

fontName

The value for `NSFontNameAttribute`.

size

The value for `NSFontSizeAttribute`.

Return Value
The new font descriptor.

Availability
Available in Mac OS X v10.3 and later.

See Also
`+ fontDescriptorWithFontAttributes:` (page 1193)
`+ fontDescriptorWithName:matrix:` (page 1193)

Declared In
`NSFontDescriptor.h`

Instance Methods

fontAttributes

Returns the receiver's dictionary of attributes.

```
- (NSDictionary *)fontAttributes
```

Return Value
The attribute dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSFontDescriptor.h

fontDescriptorByAddingAttributes:

Returns a new font descriptor that is the same as the receiver but with the specified attributes taking precedence over the existing ones.

```
- (NSFontDescriptor *)fontDescriptorByAddingAttributes:(NSDictionary *)attributes
```

Parameters
attributes
The new attributes.

Return Value
The new font descriptor.

Availability
Available in Mac OS X v10.4 and later.

See Also
+ fontDescriptorWithFontAttributes: (page 1193)

Declared In
NSFontDescriptor.h

fontDescriptorWithFace:

Returns a new font descriptor that is the same as the receiver but with the specified face.

```
- (NSFontDescriptor *)fontDescriptorWithFace:(NSString *)newFace
```

Parameters
newFace
The new font face.

Return Value
The new font descriptor.

Availability
Available in Mac OS X v10.4 and later.

See Also
+ fontDescriptorWithFontAttributes: (page 1193)

Declared In
NSFontDescriptor.h

fontDescriptorWithFamily:

Returns a new font descriptor whose attributes are the same as the receiver but from the specified family.

```
- (NSFontDescriptor *)fontDescriptorWithFamily:(NSString *)newFamily
```

Parameters

newFamily
The new font family.

Return Value
The new font descriptor.

Availability
Available in Mac OS X v10.4 and later.

See Also
+ fontDescriptorWithFontAttributes: (page 1193)

Declared In
NSFontDescriptor.h

fontDescriptorWithMatrix:

Returns a new font descriptor that is the same as the receiver but with the specified matrix.

```
- (NSFontDescriptor *)fontDescriptorWithMatrix:(NSAffineTransform *)matrix
```

Parameters

matrix
The new font matrix.

Return Value
The new font descriptor.

Availability
Available in Mac OS X v10.4 and later.

See Also
+ fontDescriptorWithFontAttributes: (page 1193)

Declared In
NSFontDescriptor.h

fontDescriptorWithSize:

Returns a new font descriptor that is the same as the receiver but with the specified point size.

```
- (NSFontDescriptor *)fontDescriptorWithSize:(CGFloat)newPointSize
```

Parameters

newPointSize

The new point size.

Return Value

The new font descriptor.

Availability

Available in Mac OS X v10.4 and later.

See Also

+ fontDescriptorWithFontAttributes: (page 1193)

Declared In

NSFontDescriptor.h

fontDescriptorWithSymbolicTraits:

Returns a new font descriptor that is the same as the receiver but with the specified symbolic traits taking precedence over the existing ones.

```
- (NSFontDescriptor
    *)fontDescriptorWithSymbolicTraits:(NSFontSymbolicTraits)symbolicTraits
```

Parameters

symbolicTraits

The new symbolic traits.

Return Value

The new font descriptor.

Availability

Available in Mac OS X v10.4 and later.

See Also

+ fontDescriptorWithFontAttributes: (page 1193)

Declared In

NSFontDescriptor.h

initWithFontAttributes:

Initializes and returns a new font descriptor with the specified attributes.

```
- (id)initWithFontAttributes:(NSDictionary *)attributes
```

Parameters

attributes

The attributes for the new font descriptor. If nil, the font descriptor's attribute dictionary will be empty.

Return Value

The new font descriptor.

Availability
Available in Mac OS X v10.3 and later.

See Also
`+ fontDescriptorWithFontAttributes:` (page 1193)

Declared In
`NSFontDescriptor.h`

matchingFontDescriptorsWithMandatoryKeys:

Returns all the fonts available on the system whose specified attributes match those of the receiver.

```
- (NSArray *)matchingFontDescriptorsWithMandatoryKeys:(NSSet *)mandatoryKeys
```

Parameters

mandatoryKeys
Keys that must be identical to be matched. Can be `nil`.

Return Value
The matching font descriptors.

Discussion
For example, suppose there are two versions of a given font installed that differ in the number of glyphs covered (the new version has more glyphs). If you explicitly specify `NSFontNameAttribute` (page 1201) as the only mandatory key, then a font descriptor that specifies a font name and character set by default matches both versions, since the character set attribute is not used for matching. If you specify that font name and character set keys are mandatory, the returned array contains only the font that matches both keys.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSFontDescriptor.h`

matchingFontDescriptorWithMandatoryKeys:

Returns a normalized font descriptor whose specified attributes match those of the receiver.

```
- (NSFontDescriptor *)matchingFontDescriptorWithMandatoryKeys:(NSSet *)mandatoryKeys
```

Parameters

mandatoryKeys
Keys that must be identical to be matched. Can be `nil`.

Return Value
The matching font descriptor.

Discussion
The returned font descriptor is the first element returned from `matchingFontDescriptorsWithMandatoryKeys:` (page 1198).

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSFontDescriptor.h

matrix

Returns the current transform matrix of the receiver.

```
- (NSAffineTransform *)matrix
```

Return Value
The transform matrix.

Availability
Available in Mac OS X v10.4 and later.

See Also
- pointSize (page 1199)

Declared In
NSFontDescriptor.h

objectForKey:

Returns the font attribute specified by the given key.

```
- (id)objectForKey:(NSString *)anAttribute
```

Parameters
anAttribute
The font attribute key.

Return Value
The font attribute corresponding to *anAttribute*. For valid values of *anAttribute*, see "Font attributes" (page 1201).

Availability
Available in Mac OS X v10.4 and later.

See Also
+ fontDescriptorWithFontAttributes: (page 1193)
- fontAttributes (page 1194)
- symbolicTraits (page 1200)

Declared In
NSFontDescriptor.h

pointSize

Returns the point size of the receiver.

```
- (CGFloat)pointSize
```

Return Value
The receiver's point size.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `fontAttributes` (page 1194)
- `matrix` (page 1199)

Declared In
`NSFontDescriptor.h`

postscriptName

Returns the PostScript name of the receiver.

```
- (NSString *)postscriptName
```

Return Value
The receiver's Postscript name.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `fontAttributes` (page 1194)
- `symbolicTraits` (page 1200)

Declared In
`NSFontDescriptor.h`

symbolicTraits

Returns a bit mask that describes the traits of the receiver.

```
- (NSFontSymbolicTraits)symbolicTraits
```

Return Value
The receiver's font traits.

Discussion
The traits describe the font's characteristics—see `NSFontSymbolicTraits` (page 1205).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `fontAttributes` (page 1194)
- `postscriptName` (page 1200)

Declared In
`NSFontDescriptor.h`

Constants

Font Attributes

These font attributes are defined by `NSFontDescriptor`.

```
NSString *NSFontFamilyAttribute;
NSString *NSFontNameAttribute;
NSString *NSFontFaceAttribute;
NSString *NSFontSizeAttribute;
NSString *NSFontVisibleNameAttribute;
NSString *NSFontColorAttribute;
NSString *NSFontMatrixAttribute;
NSString *NSFontVariationAttribute;
NSString *NSFontCharacterSetAttribute;
NSString *NSFontCascadeListAttribute;
NSString *NSFontTraitsAttribute;
NSString *NSFontFixedAdvanceAttribute;
NSString *NSFontFeatureSettingsAttribute
```

Constants

`NSFontFamilyAttribute`

An optional `NSString` object that specifies the font family.

Available in Mac OS X v10.3 and later.

Declared in `NSFontDescriptor.h`.

`NSFontNameAttribute`

An optional `NSString` object that specifies the font name.

Available in Mac OS X v10.3 and later.

Declared in `NSFontDescriptor.h`.

`NSFontFaceAttribute`

An optional `NSString` object that specifies the font face.

Available in Mac OS X v10.3 and later.

Declared in `NSFontDescriptor.h`.

`NSFontSizeAttribute`

An optional `NSString` object, containing a float value, that specifies the font size.

Available in Mac OS X v10.3 and later.

Declared in `NSFontDescriptor.h`.

`NSFontVisibleNameAttribute`

An optional `NSString` object that specifies the font's visible name.

Available in Mac OS X v10.3 and later.

Declared in `NSFontDescriptor.h`.

`NSFontColorAttribute`

An optional `NSData` object that specifies the font color. (Deprecated. Use `NSForegroundColorAttributeName` (page 271) instead.)

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.4.

Declared in `NSFontDescriptor.h`.

`NSFontMatrixAttribute`

An `NSAffineTransform` instance that specifies the font's transformation matrix.

The default value is the identity matrix.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontVariationAttribute`

An `NSDictionary` instance that describes the font's variation axis.

The default value is supplied by the font. See "Font variation axis dictionary keys" (page 1204) for dictionary keys.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontCharacterSetAttribute`

An `NSCharacterSet` instance that represents the set of Unicode characters covered by the font.

The default value is supplied by the font.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontCascadeListAttribute`

An `NSArray` instance—each member of the array is a sub-descriptor.

The default value is the system default cascading list for user's locale.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontTraitsAttribute`

An `NSDictionary` instance instance fully describing font traits.

The default value is supplied by the font. See "Font traits dictionary keys" (page 1203) for dictionary keys.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontFixedAdvanceAttribute`

An `NSNumber` instance containing a float value that overrides the glyph advancement specified by the font.

The default value is `0.0`.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontFeatureSettingsAttribute`
An array of dictionaries representing non-default font feature settings.

Each dictionary contains `NSFontFeatureTypeIdentifierKey` (page 1205) and `NSFontFeatureSelectorIdentifierKey` (page 1205).

Available in Mac OS X v10.5 and later.

Declared in `NSFontDescriptor.h`.

Discussion
You can retrieve the values for these attributes using `objectForKey:` (page 1199).

Declared In
`NSFontDescriptor.h`

Font Traits Dictionary Keys

The following constants can be used as keys to retrieve information about a font descriptor from its trait dictionary.

```
NSString *NSFontSymbolicTrait;
NSString *NSFontWeightTrait;
NSString *NSFontWidthTrait;
NSString *NSFontSlantTrait;
```

Constants
`NSFontSymbolicTrait`
The symbolic traits value as an `NSNumber` object.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontWeightTrait`
The normalized weight value as an `NSNumber` object.

The valid value range is from `-1.0` to `1.0`. The value of `0.0` corresponds to the regular or medium font weight.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontWidthTrait`
The relative inter-glyph spacing value as an `NSNumber` object.

The valid value range is from `-1.0` to `1.0`. The value of `0.0` corresponds to the regular glyph spacing.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontSlantTrait`
The relative slant angle value as an `NSNumber` object.

The valid value range is from `-1.0` to `1.0`. The value of `0.0` corresponds to `0` degree clockwise rotation from the vertical and `1.0` corresponds to `30` degrees clockwise rotation.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

Discussion
These keys are used with `NSFontTraitsAttribute` (page 1202).

Declared In
NSFontDescriptor.h

Font Variation Axis Dictionary Keys

The following constants can be used as keys to retrieve information about a font descriptor from its variation axis dictionary.

```
NSString *NSFontVariationAxisIdentifierKey;
NSString *NSFontVariationAxisMinimumValueKey;
NSString *NSFontVariationAxisMaximumValueKey;
NSString *NSFontVariationAxisDefaultValueKey;
NSString *NSFontVariationAxisNameKey;
```

Constants

`NSFontVariationAxisIdentifierKey`
The axis identifier value as an `NSNumber` object.
Available in Mac OS X v10.4 and later.
Declared in `NSFontDescriptor.h`.

`NSFontVariationAxisMinimumValueKey`
The minimum axis value as an `NSNumber` object.
Available in Mac OS X v10.4 and later.
Declared in `NSFontDescriptor.h`.

`NSFontVariationAxisMaximumValueKey`
The maximum axis value as an `NSNumber` object.
Available in Mac OS X v10.4 and later.
Declared in `NSFontDescriptor.h`.

`NSFontVariationAxisDefaultValueKey`
The default axis value as an `NSNumber` object.
Available in Mac OS X v10.4 and later.
Declared in `NSFontDescriptor.h`.

`NSFontVariationAxisNameKey`
The localized variation axis name.
Available in Mac OS X v10.4 and later.
Declared in `NSFontDescriptor.h`.

Discussion
These keys are used with `NSFontVariationAttribute` (page 1202).

Declared In
NSFontDescriptor.h

Font Feature Keys

The following constants can be used as keys to retrieve information about a font descriptor from its feature dictionary.

```
NSString *NSFontFeatureTypeIdentifierKey;
NSString *NSFontFeatureSelectorIdentifierKey;
```

Constants

`NSFontFeatureTypeIdentifierKey`

An `NSNumber` object specifying a font feature type such as ligature, character shape, and so on. See "Font Features" in *ATSUI Programming Guide* for predefined feature types.

Available in Mac OS X v10.5 and later.

Declared in `NSFontDescriptor.h`.

`NSFontFeatureSelectorIdentifierKey`

An `NSNumber` object specifying a font feature selector such as common ligature off, traditional character shape, and so on. See "Font Features" in *ATSUI Programming Guide* for predefined feature selectors.

Available in Mac OS X v10.5 and later.

Declared in `NSFontDescriptor.h`.

Discussion

These keys are used with `NSFontFeatureSettingsAttribute` (page 1203).

Declared In

`NSFontDescriptor.h`

NSFontSymbolicTraits

`NSFontSymbolicTraits` symbolically describes stylistic aspects of a font.

```
typedef uint32_t NSFontSymbolicTraits;
```

Discussion

The upper 16 bits is used to describe appearance of the font (see `NSFontFamilyClass` (page 1205)) whereas the lower 16 bits is used for typeface information (see `Typeface information` (page 1207)). The font appearance information represented by the upper 16 bits can be used for stylistic font matching. The symbolic traits supersede the existing `NSFontTraitMask` type used by `NSFontManager`. The corresponding values are kept compatible between `NSFontTraitMask` and `NSFontSymbolicTraits`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSFontDescriptor.h`

NSFontFamilyClass

These constants classify certain stylistic qualities of the font. These values correspond closely to the font class values in the OpenType OS/2 table. The class values are bundled in the upper four bits of the `NSFontSymbolicTraits` and can be accessed via `NSFontFamilyClassMask`. For more information about the specific meaning of each identifier, refer to the OpenType specification.

```
enum {
    NSFontUnknownClass = 0 << 28,
    NSFontOldStyleSerifsClass = 1 << 28,
    NSFontTransitionalSerifsClass = 2 << 28,
    NSFontModernSerifsClass = 3 << 28,
    NSFontClarendonSerifsClass = 4 << 28,
    NSFontSlabSerifsClass = 5 << 28,
    NSFontFreeformSerifsClass = 7 << 28,
    NSFontSansSerifClass = 8 << 28,
    NSFontOrnamentalsClass = 9 << 28,
    NSFontScriptsClass = 10 << 28,
    NSFontSymbolicClass = 12 << 28
};
typedef uint32_t NSFontFamilyClass;
```

Constants

`NSFontUnknownClass`

The font has no design classification.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontOldStyleSerifsClass`

The font's style is based on the Latin printing style of the 15th to 17th century.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontTransitionalSerifsClass`

The font's style is based on the Latin printing style of the 18th to 19th century.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontModernSerifsClass`

The font's style is based on the Latin printing style of the 20th century.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontClarendonSerifsClass`

The font's style is a variation of the Oldstyle Serifs and the Transitional Serifs.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontSlabSerifsClass`

The font's style is characterized by serifs with a square transition between the strokes and the serifs (no brackets).

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontFreeformSerifsClass`

The font's style includes serifs, but it expresses a design freedom that does not generally fit within the other serif design classifications.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontSansSerifClass`
: The font's style includes most basic letter forms (excluding Scripts and Ornamentals) that do not have serifs on the strokes.

  Available in Mac OS X v10.4 and later.

  Declared in `NSFontDescriptor.h`.

`NSFontOrnamentalsClass`
: The font's style includes highly decorated or stylized character shapes such as those typically used in headlines.

  Available in Mac OS X v10.4 and later.

  Declared in `NSFontDescriptor.h`.

`NSFontScriptsClass`
: The font's style is among those typefaces designed to simulate handwriting.

  Available in Mac OS X v10.4 and later.

  Declared in `NSFontDescriptor.h`.

`NSFontSymbolicClass`
: The font's style is generally design independent, making it suitable for special characters (icons, dingbats, technical symbols, and so on) that may be used equally well with any font.

  Available in Mac OS X v10.4 and later.

  Declared in `NSFontDescriptor.h`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSFontDescriptor.h`

NSFontFamilyClassMask

This constant is used to access `NSFontFamilyClass` values in the upper four bits of `NSFontSymbolicTraits`.

```
enum {
    NSFontFamilyClassMask = 0xF0000000
};
```

Constants
`NSFontFamilyClassMask`
: The font family class mask used to access `NSFontFamilyClass` values.

  Available in Mac OS X v10.4 and later.

  Declared in `NSFontDescriptor.h`.

Typeface Information

Typeface information is specified by the lower 16 bits of `NSFontSymbolicTraits` using the following constants.

```
enum {
    NSFontItalicTrait = (1 << 0),
    NSFontBoldTrait = (1 << 1),
    NSFontExpandedTrait = (1 << 5),
    NSFontCondensedTrait = (1 << 6),
    NSFontMonoSpaceTrait = (1 << 10),
    NSFontVerticalTrait = (1 << 11),
    NSFontUIOptimizedTrait = (1 << 12)
};
```

Constants

`NSFontItalicTrait`

The font's typestyle is italic.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontBoldTrait`

The font's typestyle is boldface.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontExpandedTrait`

The font's typestyle is expanded. Expanded and condensed traits are mutually exclusive.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontCondensedTrait`

The font's typestyle is condensed. Expanded and condensed traits are mutually exclusive.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontMonoSpaceTrait`

The font uses fixed-pitch glyphs if available. The font may have multiple glyph advances (many CJK glyphs contain two spaces).

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontVerticalTrait`

The font uses vertical glyph variants and metrics.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

`NSFontUIOptimizedTrait`

The font synthesizes appropriate attributes for user interface rendering, such as control titles, if necessary.

Available in Mac OS X v10.4 and later.

Declared in `NSFontDescriptor.h`.

Declared In

`NSFontDescriptor.h`

NSFontManager Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSFontManager.h
Companion guide	Font Panel
Related sample code	CoreTextArcCocoa NewsReader QTKitTimeCode Quartz Composer WWDC 2005 TextEdit ToolbarSample

Overview

`NSFontManager` is the center of activity for the font conversion system. It records the currently selected font, updates the Font panel and Font menu to reflect the selected font, initiates font changes, and converts fonts in response to requests from text-bearing objects. In a more prosaic role, `NSFontManager` can be queried for the fonts available to the application and for the particular attributes of a font, such as whether it's condensed or extended.

You normally set up a font manager and the Font menu using Interface Builder. However, you can also do so programmatically by getting the shared font manager instance and having it create the standard Font menu at runtime:

```
NSFontManager *fontManager = [NSFontManager sharedFontManager];
NSMenu *fontMenu = [fontManager fontMenu:YES];
```

You can then add the Font menu to your application's main menu. Once the Font menu is installed, your application automatically gains the functionality of both the Font menu and the Font panel.

As of Mac OS X version 10.3, font collections are managed by `NSFontManager`.

Tasks

Getting the Shared Font Manager

`+ sharedFontManager` (page 1215)
Returns the shared instance of the font manager for the application, creating it if necessary.

Changing the Default Font Conversion Classes

`+ setFontManagerFactory:` (page 1214)
Sets the class object used to create the font manager to the given class.

`+ setFontPanelFactory:` (page 1214)
Sets the class used to create the Font panel to the given class.

Getting Available Fonts

`- availableFonts` (page 1218)
Returns the names of the fonts available in the system (not the `NSFont` objects themselves).

`- availableFontFamilies` (page 1217)
Returns the names of the font families available in the system.

`- availableFontNamesWithTraits:` (page 1218)
Returns the names of the fonts available in the system whose traits are described exactly by the given font trait mask (not the `NSFont` objects themselves).

`- availableMembersOfFontFamily:` (page 1219)
Returns an array with one entry for each available member of a font family.

Setting and Examining the Selected Font

`- setSelectedFont:isMultiple:` (page 1237)
Records the given font as the currently selected font and updates the Font panel to reflect this.

`- selectedFont` (page 1234)
Returns the last font recorded.

`- isMultiple` (page 1230)
Indicates whether the last font selection recorded has multiple fonts.

`- sendAction` (page 1234)
Sends the receiver's action message up the responder chain, initiating a font change for whatever conversion and trait to change were last requested.

`- localizedNameForFamily:face:` (page 1230)
Returns a localized string with the name of the specified font family and face, if one exists.

Sending Action Methods

- `addFontTrait:` (page 1216)
  This action method causes the receiver to send its action message up the responder chain.
- `removeFontTrait:` (page 1233)
  This action method causes the receiver to send its action message up the responder chain.
- `modifyFont:` (page 1230)
  This action method causes the receiver to send its action message up the responder chain.
- `modifyFontViaPanel:` (page 1231)
  This action method causes the receiver to send its action message up the responder chain.
- `orderFrontStylesPanel:` (page 1232)
  This action method opens the Font styles panel.
- `orderFrontFontPanel:` (page 1232)
  This action method opens the Font panel by sending it an `orderFront:` (page 3351) message, creating the Font panel if necessary.

Converting Fonts Automatically

- `convertFont:` (page 1220)
  Converts the given font according to the object that initiated a font change, typically the Font panel or Font menu.
- `changeFont:` (page 1239) *delegate method*
  Informs responders of a font change.

Converting Fonts Manually

- `convertFont:toFace:` (page 1221)
  Returns a font whose traits are as similar as possible to those of the given font except for the typeface, which is changed to the given typeface.
- `convertFont:toFamily:` (page 1222)
  Returns a font whose traits are as similar as possible to those of the given font except for the font family, which is changed to the given family.
- `convertFont:toHaveTrait:` (page 1222)
  Returns a font whose traits are the same as those of the given font, except that the traits are changed to include the single specified trait.
- `convertFont:toNotHaveTrait:` (page 1223)
  Returns an `NSFont` object with the same traits as the given font, except for the traits in the given font trait mask, which are removed.
- `convertFont:toSize:` (page 1224)
  Returns an `NSFont` object whose traits are the same as those of the given font, except for the size, which is changed to the given size.
- `convertWeight:ofFont:` (page 1225)
  Returns an `NSFont` object whose weight is greater or lesser than that of the given font, if possible.
- `currentFontAction` (page 1226)
  Returns the current font conversion action.

- `convertFontTraits:` (page 1224)
  Converts font traits to a new traits mask value.

Getting a Particular Font

- `fontWithFamily:traits:weight:size:` (page 1229)
  Attempts to load a font with the specified characteristics.

Examining Fonts

- `traitsOfFont:` (page 1238)
  Returns the traits of the given font.
- `fontNamed:hasTraits:` (page 1228)
  Indicates whether the given font has all the specified traits.
- `weightOfFont:` (page 1239)
  Returns a rough numeric measure the weight of the given font.

Managing the Font Panel and Font Menu

- `setEnabled:` (page 1236)
  Controls whether the font conversion system's user interface items (the Font panel and Font menu items) are enabled.
- `isEnabled` (page 1229)
  Indicates whether the font conversion system's user interface items (the Font panel and Font menu items) are enabled.
- `fontManager:willIncludeFont:` (page 1240) *delegate method*
  Requests permission from the Font panel delegate to display the given font name in the Font panel.
- `fontPanel:` (page 1228)
  Returns the application's shared Font panel object, optionally creating it if necessary.
- `setFontMenu:` (page 1236)
  Records the given menu as the application's Font menu.
- `fontMenu:` (page 1227)
  Returns the menu that's hooked up to the font conversion system, optionally creating it if necessary.

Setting the Delegate

- `setDelegate:` (page 1235)
  Sets the receiver's delegate to the given object.
- `delegate` (page 1226)
  Returns the receiver's delegate.

Accessing the Action Method

- `setAction:` (page 1235)
  Sets the action that's sent to the first responder, when the user selects a new font from the Font panel or chooses a command from the Font menu, to the given selector.
- `action` (page 1215)
  Returns the action sent to the first responder when the user selects a new font from the Font panel or chooses a command from the Font menu.
- `setTarget:` (page 1238)
  Sets the target for the `sendAction` (page 1234) method.
- `target` (page 1238)
  Returns the target for the `sendAction` (page 1234) method.

Setting Attributes

- `setSelectedAttributes:isMultiple:` (page 1236)
  Informs the paragraph and character formatting panels when text in a selection has changed attributes.
- `convertAttributes:` (page 1220)
  Converts attributes in response to an object initiating an attribute change, typically the Font panel or Font menu.

Working with Font Descriptors

- `availableFontNamesMatchingFontDescriptor:` (page 1217)
  Returns the names of the fonts that match the attributes in the given font descriptor.
- `collectionNames` (page 1219)
  Returns the names of the currently loaded font collections.
- `fontDescriptorsInCollection:` (page 1227)
  Returns an array of the font descriptors in the collection specified by the given collection name.
- `addCollection:options:` (page 1216)
  Adds a specified font collection to the font manager with a given set of options.
- `removeCollection:` (page 1233)
  Removes the specified font collection.
- `addFontDescriptors:toCollection:` (page 1216)
  Adds an array of font descriptors to the specified font collection.
- `removeFontDescriptor:fromCollection:` (page 1233)
  Removes the specified font descriptor from the specified collection.

Class Methods

setFontManagerFactory:

Sets the class object used to create the font manager to the given class.

```
+ (void)setFontManagerFactory:(Class)aClass
```

Parameters

aClass

The new font manager factory class, which should be a subclass of `NSFontManager`.

Discussion

When the `NSFontManager` class object receives a `sharedFontManager` (page 1215) message, it creates an instance of *aClass*, if no instance already exists. Your font manager class should implement `init` as its designated initializer. The default font manager factory is `NSFontManager`.

This method must be invoked before your application's main nib file is loaded, such as in the application delegate's `applicationWillFinishLaunching:` method.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ setFontPanelFactory:` (page 1214)

Declared In

`NSFontManager.h`

setFontPanelFactory:

Sets the class used to create the Font panel to the given class.

```
+ (void)setFontPanelFactory:(Class)factoryId
```

Parameters

factoryId

The new font panel factory class, which should be a subclass of `NSFontPanel`.

Discussion

Invoke this method before accessing the Font panel in any way, such as in the application delegate's `applicationWillFinishLaunching:` method.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ setFontManagerFactory:` (page 1214)

Declared In

`NSFontManager.h`

sharedFontManager

Returns the shared instance of the font manager for the application, creating it if necessary.

```
+ (NSFontManager *)sharedFontManager
```

Return Value
The shared font manager.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ setFontManagerFactory:` (page 1214)

Related Sample Code
CoreTextArcCocoa
NewsReader
QTKitTimeCode
Quartz Composer WWDC 2005 TextEdit
ToolbarSample

Declared In
`NSFontManager.h`

Instance Methods

action

Returns the action sent to the first responder when the user selects a new font from the Font panel or chooses a command from the Font menu.

```
- (SEL)action
```

Return Value
The selector for the action.

Discussion
The default action is `changeFont:` (page 1239).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAction:` (page 1235)

Declared In
`NSFontManager.h`

addCollection:options:

Adds a specified font collection to the font manager with a given set of options.

```
- (BOOL)addCollection:(NSString *)collectionName options:(NSInteger)collectionOptions
```

Parameters

collectionName

The collection to add.

collectionOptions

The option described in "Font Collection Mask" (page 1241). This option is not yet implemented.

Return Value

YES if the font collection was successfully added; otherwise, NO.

Availability

Available in Mac OS X v10.3 and later.

See Also

- removeCollection: (page 1233)

Declared In

NSFontManager.h

addFontDescriptors:toCollection:

Adds an array of font descriptors to the specified font collection.

```
- (void)addFontDescriptors:(NSArray *)descriptors toCollection:(NSString
    *)collectionName
```

Parameters

descriptors

The font descriptors to add.

collectionName

The font collection to which descriptors are added.

Availability

Available in Mac OS X v10.3 and later.

See Also

- removeFontDescriptor:fromCollection: (page 1233)

Declared In

NSFontManager.h

addFontTrait:

This action method causes the receiver to send its action message up the responder chain.

```
- (void)addFontTrait:(id)sender
```

Parameters

sender

The control that sent the message.

Discussion

By default, the action message is `changeFont:` (page 1239).

When a responder replies by providing a font to convert in a `convertFont:` (page 1220) message, the receiver converts the font by adding the trait specified by *sender*. This trait is determined by sending a `tag` message to *sender* and interpreting it as a font trait mask for a `convertFont:toHaveTrait:` (page 1222) message.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `removeFontTrait:` (page 1233)
- `modifyFont:` (page 1230)
- `modifyFontViaPanel:` (page 1231)

Declared In

`NSFontManager.h`

availableFontFamilies

Returns the names of the font families available in the system.

```
- (NSArray *)availableFontFamilies
```

Return Value

The names of the available font families.

Discussion

These fonts are in various system font directories.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `availableFontNamesWithTraits:` (page 1218)
- `availableFonts` (page 1218)

Declared In

`NSFontManager.h`

availableFontNamesMatchingFontDescriptor:

Returns the names of the fonts that match the attributes in the given font descriptor.

```
- (NSArray *)availableFontNamesMatchingFontDescriptor:(NSFontDescriptor *)descriptor
```

Parameters

descriptor

The font descriptor whose attributes are matched.

Return Value
The names of the matching fonts.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSFontManager.h

availableFontNamesWithTraits:

Returns the names of the fonts available in the system whose traits are described exactly by the given font trait mask (not the NSFont objects themselves).

```
- (NSArray *)availableFontNamesWithTraits:(NSFontTraitMask)fontTraitMask
```

Parameters

fontTraitMask

The font traits for which to return font names. You specify the desired traits by combining the font trait mask values described in "Constants" (page 1241) using the C bitwise OR operator.

Return Value
The names of the corresponding fonts.

Discussion
These fonts are in various system font directories.

If *fontTraitMask* is 0, this method returns all fonts that are neither italic nor bold. This result is the same one you'd get if *fontTraitMask* were NSUnitalicFontMask | NSUnboldFontMask.

Availability
Available in Mac OS X v10.0 and later.

See Also
- availableFontFamilies (page 1217)
- availableFonts (page 1218)

Declared In
NSFontManager.h

availableFonts

Returns the names of the fonts available in the system (not the NSFont objects themselves).

```
- (NSArray *)availableFonts
```

Return Value
The names of the available fonts.

Discussion
These fonts are in various system font directories.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `availableFontFamilies` (page 1217)
- `availableFontNamesWithTraits:` (page 1218)

Related Sample Code
NSFontAttributeExplorer

Declared In
`NSFontManager.h`

availableMembersOfFontFamily:

Returns an array with one entry for each available member of a font family.

```
- (NSArray *)availableMembersOfFontFamily:(NSString *)family
```

Parameters

family
The name of a font family, like one that `availableFontFamilies` (page 1217) returns.

Return Value
The available members of *family*. See the following discussion for a specific description.

Discussion
Each entry of the returned `NSArray` is another `NSArray` with four members, as follows:

0. The PostScript font name, as an `NSString` object.
1. The part of the font name used in the font panel that's not the font name, as an `NSString` object. This value is not localized—for example, `"Roman"`, `"Italic"`, or `"Bold"`.
2. The font's weight, as an `NSNumber`.
3. The font's traits, as an `NSNumber`.

The members of the family are arranged in the font panel order (narrowest to widest, lightest to boldest, plain to italic).

For example, if you call `availableMembersOfFontFamily:@"Times"`, it might return an array like this:

```
(("Times-Roman", "Roman", 5, 4),
 ("Times-Italic", "Italic", 6, 5),
 ("Times-Bold", "Bold", 9, 2),
 ("Times-BoldItalic", "Bold Italic", 9, 3)
)
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFontManager.h`

collectionNames

Returns the names of the currently loaded font collections.

```
- (NSArray *)collectionNames
```

Return Value
The names of the current font collections.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `fontDescriptorsInCollection:` (page 1227)

Declared In
`NSFontManager.h`

convertAttributes:

Converts attributes in response to an object initiating an attribute change, typically the Font panel or Font menu.

```
- (NSDictionary *)convertAttributes:(NSDictionary *)attributes
```

Parameters
attributes
The current attributes.

Return Value
The converted attributes, or *attributes* itself if the conversion isn't possible.

Discussion
Attributes unused by the sender should not be changed or removed.

This method is usually invoked on the sender of `changeAttributes:` (page 2883). See NSTextView for more information.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setSelectedAttributes:isMultiple:` (page 1236)

Declared In
`NSFontManager.h`

convertFont:

Converts the given font according to the object that initiated a font change, typically the Font panel or Font menu.

```
- (NSFont *)convertFont:(NSFont *)aFont
```

Parameters
aFont
The font to convert.

Return Value
The converted font, or `aFont` itself if the conversion isn't possible.

Discussion
This method is invoked in response to an action message such as `addFontTrait:` (page 1216) or `modifyFontViaPanel:` (page 1231). These initiating methods cause the font manager to query the sender for the action to take and the traits to change. See "Converting Fonts Manually" (page 1211) for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `convertFont:toFace:` (page 1221)
- `convertFont:toFamily:` (page 1222)
- `convertFont:toHaveTrait:` (page 1222)
- `convertFont:toNotHaveTrait:` (page 1223)
- `convertFont:toSize:` (page 1224)
- `convertWeight:ofFont:` (page 1225)

Declared In
`NSFontManager.h`

convertFont:toFace:

Returns a font whose traits are as similar as possible to those of the given font except for the typeface, which is changed to the given typeface.

```
- (NSFont *)convertFont:(NSFont *)aFont toFace:(NSString *)typeface
```

Parameters

`aFont`
The font whose traits are matched.

`typeface`
The new typeface; a fully specified family-face name, such as Helvetica-BoldOblique or Times-Roman.

Return Value
A font with matching traits and the given typeface, or `aFont` if it can't be converted.

Discussion
This method attempts to match the weight and posture of `aFont` as closely as possible. Italic is mapped to Oblique, for example. Weights are mapped based on an approximate numeric scale of 0 to 15.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `convertFont:toFamily:` (page 1222)
- `convertFont:toHaveTrait:` (page 1222)
- `convertFont:toNotHaveTrait:` (page 1223)
- `convertFont:toSize:` (page 1224)
- `convertWeight:ofFont:` (page 1225)
- `convertFont:` (page 1220)

Declared In
NSFontManager.h

convertFont:toFamily:

Returns a font whose traits are as similar as possible to those of the given font except for the font family, which is changed to the given family.

```
- (NSFont *)convertFont:(NSFont *)aFont toFamily:(NSString *)family
```

Parameters

aFont
The font whose traits are matched.

family
The new font famliy; a generic font name, such as Helvetica or Times.

Return Value
A font with matching traits and the given family, or *aFont* if it can't be converted.

Discussion
This method attempts to match the weight and posture of *aFont* as closely as possible. Italic is mapped to Oblique, for example. Weights are mapped based on an approximate numeric scale of 0 to 15.

Availability
Available in Mac OS X v10.0 and later.

See Also
- convertFont:toFace: (page 1221)
- convertFont:toHaveTrait: (page 1222)
- convertFont:toNotHaveTrait: (page 1223)
- convertFont:toSize: (page 1224)
- convertWeight:ofFont: (page 1225)
- convertFont: (page 1220)

Declared In
NSFontManager.h

convertFont:toHaveTrait:

Returns a font whose traits are the same as those of the given font, except that the traits are changed to include the single specified trait.

```
- (NSFont *)convertFont:(NSFont *)aFont toHaveTrait:(NSFontTraitMask)fontTrait
```

Parameters

aFont
The font whose traits are matched.

fontTrait
The new trait; may be any one of the traits described in "Constants" (page 1241). Using NSUnboldFontMask or NSUnitalicFontMask removes the bold or italic trait, respectively.

Return Value
A font with matching traits including the given trait, or *aFont* if it can't be converted.

Discussion
Using `NSUnboldFontMask` or `NSUnitalicFontMask` removes the bold or italic trait, respectively.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `convertFont:toNotHaveTrait:` (page 1223)
- `convertFont:toFace:` (page 1221)
- `convertFont:toFamily:` (page 1222)
- `convertFont:toSize:` (page 1224)
- `convertWeight:ofFont:` (page 1225)
- `convertFont:` (page 1220)

Related Sample Code
CoreTextArcCocoa
NewsReader
ToolbarSample

Declared In
`NSFontManager.h`

convertFont:toNotHaveTrait:

Returns an `NSFont` object with the same traits as the given font, except for the traits in the given font trait mask, which are removed.

```
- (NSFont *)convertFont:(NSFont *)aFont toNotHaveTrait:(NSFontTraitMask)fontTraitMask
```

Parameters

aFont
The font whose traits are matched.

fontTraitMask
The mask for the traits to remove, created using the C bitwise OR operator to combine the traits described in "Constants" (page 1241). Using `NSUnboldFontMask` or `NSUnitalicFontMask` removes the bold or italic trait, respectively.

Return Value
A font with matching traits minus the given traits, or *aFont* if it can't be converted.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `convertFont:toHaveTrait:` (page 1222)
- `convertFont:toFace:` (page 1221)
- `convertFont:toFamily:` (page 1222)
- `convertFont:toSize:` (page 1224)
- `convertWeight:ofFont:` (page 1225)

- convertFont: (page 1220)

Related Sample Code
CoreTextArcCocoa
ToolbarSample

Declared In
NSFontManager.h

convertFont:toSize:

Returns an NSFont object whose traits are the same as those of the given font, except for the size, which is changed to the given size.

```
- (NSFont *)convertFont:(NSFont *)aFont toSize:(CGFloat)size
```

Parameters

aFont

The font whose traits are matched.

size

The new font size.

Return Value
A font with matching traits except in the new size, or *aFont* if it can't be converted.

Availability
Available in Mac OS X v10.0 and later.

See Also
- convertFont:toFace: (page 1221)
- convertFont:toFamily: (page 1222)
- convertFont:toHaveTrait: (page 1222)
- convertFont:toNotHaveTrait: (page 1223)
- convertWeight:ofFont: (page 1225)
- convertFont: (page 1220)

Related Sample Code
ToolbarSample

Declared In
NSFontManager.h

convertFontTraits:

Converts font traits to a new traits mask value.

```
- (NSFontTraitMask)convertFontTraits:(NSFontTraitMask)traits
```

Parameters

traits

The current font traits.

Return Value
The new traits mask value to be used by `convertFont:` (page 1220).

Discussion
This method is intended to be invoked to query the font traits while the action message (usually `changeFont:` (page 1239)) is being invoked when the current font action is either `NSAddTraitFontAction` (page 1244) or `NSRemoveTraitFontAction` (page 1244).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSFontManager.h`

convertWeight:ofFont:

Returns an `NSFont` object whose weight is greater or lesser than that of the given font, if possible.

```
- (NSFont *)convertWeight:(BOOL)increaseFlag ofFont:(NSFont *)aFont
```

Parameters

increaseFlag
If `YES`, a heavier font is returned; if it's `NO`, a lighter font is returned.

aFont
The font whose weight is increased or decreased.

Return Value
A font with matching traits except for the new weight, or *aFont* if it can't be converted.

Discussion
Weights are graded along the following scale. The list on the left gives Apple's terminology, and the list on the right gives the ISO equivalents. Names on the same line are treated as identical:

Apple Terminology	ISO Equivalent
1. ultralight	
2. thin	W1. ultralight
3. light, extralight	W2. extralight
4. book	W3. light
5. regular, plain, display, roman	W4. semilight
6. medium	W5. medium
7. demi, demibold	
8. semi, semibold	W6. semibold
9. bold	W7. bold
10. extra, extrabold	W8. extrabold

Apple Terminology	ISO Equivalent
11. heavy, heavyface	
12. black, super	W9. ultrabold
13. ultra, ultrablack, fat	
14. extrablack, obese, nord	

The `NSFontManager` implementation of this method refuses to convert a font's weight if it can't maintain all other traits, such as italic and condensed. You might wish to override this method to allow a looser interpretation of weight conversion.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `convertFont:toFace:` (page 1221)
- `convertFont:toFamily:` (page 1222)
- `convertFont:toHaveTrait:` (page 1222)
- `convertFont:toNotHaveTrait:` (page 1223)
- `convertFont:toSize:` (page 1224)
- `convertFont:` (page 1220)

Declared In
`NSFontManager.h`

currentFontAction

Returns the current font conversion action.

```
- (NSFontAction)currentFontAction
```

Return Value
The current font action used by the `convertFont:` (page 1220) method.

Discussion
This method is intended to be invoked to query the font conversion action while the action message (usually `changeFont:` (page 1239)) is being invoked.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSFontManager.h`

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setDelegate: (page 1235)

Declared In
NSFontManager.h

fontDescriptorsInCollection:

Returns an array of the font descriptors in the collection specified by the given collection name.

```
- (NSArray *)fontDescriptorsInCollection:(NSString *)collectionName
```

Parameters
collectionName
The font collection for which to return descriptors.

Return Value
The font descriptors.

Availability
Available in Mac OS X v10.3 and later.

See Also
- collectionNames (page 1219)

Declared In
NSFontManager.h

fontMenu:

Returns the menu that's hooked up to the font conversion system, optionally creating it if necessary.

```
- (NSMenu *)fontMenu:(BOOL)createFlag
```

Parameters
createFlag
If YES, the menu object is created if necessary; if NO, it is not.

Return Value
The font conversion system menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setFontMenu: (page 1236)

Declared In
NSFontManager.h

fontNamed:hasTraits:

Indicates whether the given font has all the specified traits.

```
- (BOOL)fontNamed:(NSString *)typeface hasTraits:(NSFontTraitMask)fontTraitMask
```

Parameters

typeface
The name of the font.

fontTraitMask
The font traits to test, specified by combining the font trait mask values described in "Constants" (page 1241) using the C bitwise OR operator.

Return Value
YES if the font named *typeface* has all the traits specified in *fontTraitMask*; NO if it doesn't.

Discussion
Using NSUnboldFontMask returns YES if the font is not bold, NO otherwise. Using NSUnitalicFontMask returns YES if the font is not italic, NO otherwise.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSFontManager.h

fontPanel:

Returns the application's shared Font panel object, optionally creating it if necessary.

```
- (NSFontPanel *)fontPanel:(BOOL)createFlag
```

Parameters

createFlag
If YES, the Font panel object is created if necessary; if NO, it is not.

Return Value
The application's shared Font panel object.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ sharedFontPanel (page 1248) (NSFontPanel)
+ sharedFontPanelExists (page 1249) (NSFontPanel)
+ setFontPanelFactory: (page 1214)

Declared In
NSFontManager.h

fontWithFamily:traits:weight:size:

Attempts to load a font with the specified characteristics.

```
- (NSFont *)fontWithFamily:(NSString *)family traits:(NSFontTraitMask)fontTraitMask
    weight:(NSInteger)weight size:(CGFloat)size
```

Parameters

family

The generic name of the desired font, such as Times or Helvetica.

fontTraitMask

The font traits, specified by combining the font trait mask values described in "Constants" (page 1241) using the C bitwise OR operator. Using `NSUnboldFontMask` or `NSUnitalicFontMask` loads a font that doesn't have either the bold or italic trait, respectively.

weight

A hint for the weight desired, on a scale of 0 to 15: a value of 5 indicates a normal or book weight, and 9 or more a bold or heavier weight. The weight is ignored if *fontTraitMask* includes `NSBoldFontMask`.

size

The point size of the desired font.

Return Value
A font with the specified characteristics if successful, or `nil` if not.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFontManager.h`

isEnabled

Indicates whether the font conversion system's user interface items (the Font panel and Font menu items) are enabled.

```
- (BOOL)isEnabled
```

Return Value
`YES` if the font conversion system's user interface items (the Font panel and Font menu items) are enabled; `NO` if they're not.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isEnabled` (page 1250) (NSFontPanel)
- `isEnabled` (NSMenuItem)
- `setEnabled:` (page 1236)

Declared In
`NSFontManager.h`

isMultiple

Indicates whether the last font selection recorded has multiple fonts.

```
- (BOOL)isMultiple
```

Return Value
`YES` if the last font selection recorded has multiple fonts; `NO` if it's a single font.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setSelectedFont:isMultiple:` (page 1237)
- `selectedFont` (page 1234)

Declared In
`NSFontManager.h`

localizedNameForFamily:face:

Returns a localized string with the name of the specified font family and face, if one exists.

```
- (NSString *)localizedNameForFamily:(NSString *)family face:(NSString *)face
```

Parameters
family
The font family, for example, `@"Times"`.

face
The font face, for example, `@"Roman"`.

Return Value
A localized string with the name of the specified font family and face, or, if *face* is `nil`, the font family only.

Discussion
The user's locale is determined from the user's `NSLanguages` default setting. The method also loads the localized strings for the font, if they aren't already loaded.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFontManager.h`

modifyFont:

This action method causes the receiver to send its action message up the responder chain.

```
- (void)modifyFont:(id)sender
```

Parameters
sender
The control that sent the message.

Discussion
By default, the action message is `changeFont:` (page 1239).

When a responder replies by providing a font to convert in a `convertFont:` (page 1220) message, the receiver converts the font in the manner specified by *sender*. The conversion is determined by sending a `tag` message to *sender* and invoking a corresponding method:

Sender's Tag	Method Used
`NSNoFontChangeAction`	None; the font is returned unchanged.
`NSViaPanelFontAction`	The Font panel's `panelConvertFont:` (page 1250).
`NSAddTraitFontAction`	`convertFont:toHaveTrait:` (page 1222).
`NSRemoveTraitFontAction`	`convertFont:toNotHaveTrait:` (page 1223).
`NSSizeUpFontAction`	`convertFont:toSize:` (page 1224).
`NSSizeDownFontAction`	`convertFont:toSize:` (page 1224).
`NSHeavierFontAction`	`convertWeight:ofFont:` (page 1225).
`NSLighterFontAction`	`convertWeight:ofFont:` (page 1225).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addFontTrait:` (page 1216)
- `removeFontTrait:` (page 1233)
- `modifyFontViaPanel:` (page 1231)

Declared In
`NSFontManager.h`

modifyFontViaPanel:

This action method causes the receiver to send its action message up the responder chain.

```
- (void)modifyFontViaPanel:(id)sender
```

Parameters
sender
The control that sent the message.

Discussion
By default, the action message is `changeFont:` (page 1239).

When a responder replies by providing a font to convert in a `convertFont:` (page 1220) message, the receiver converts the font by sending a `panelConvertFont:` (page 1250) message to the Font panel. The panel in turn may send `convertFont:toFamily:` (page 1222), `convertFont:toHaveTrait:` (page 1222), and other specific conversion methods to make its change.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addFontTrait:` (page 1216)
- `removeFontTrait:` (page 1233)
- `modifyFont:` (page 1230)

Declared In
`NSFontManager.h`

orderFrontFontPanel:

This action method opens the Font panel by sending it an `orderFront:` (page 3351) message, creating the Font panel if necessary.

```
- (void)orderFrontFontPanel:(id)sender
```

Parameters

sender
The control that sent the message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `fontPanel:` (page 1228)
+ `setFontPanelFactory:` (page 1214)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSFontManager.h`

orderFrontStylesPanel:

This action method opens the Font styles panel.

```
- (void)orderFrontStylesPanel:(id)sender
```

Parameters

sender
The control that sent the message.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSFontManager.h`

removeCollection:

Removes the specified font collection.

```
- (BOOL)removeCollection:(NSString *)collectionName
```

Parameters

collectionName

The collection to remove.

Return Value

`YES` if the font collection was successfully removed; otherwise, `NO`.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- addCollection:options:` (page 1216)

Declared In

`NSFontManager.h`

removeFontDescriptor:fromCollection:

Removes the specified font descriptor from the specified collection.

```
- (void)removeFontDescriptor:(NSFontDescriptor *)descriptor fromCollection:(NSString
    *)collection
```

Parameters

descriptor

The font descriptor to remove.

collection

The font collection from which to remove the descriptor.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- addFontDescriptors:toCollection:` (page 1216)

Declared In

`NSFontManager.h`

removeFontTrait:

This action method causes the receiver to send its action message up the responder chain.

```
- (void)removeFontTrait:(id)sender
```

Parameters

sender

The control that sent the message.

Discussion
By default, the action message is `changeFont:` (page 1239).

When a responder replies by providing a font to convert in a `convertFont:` (page 1220) message, the receiver converts the font by removing the trait specified by *sender*. This trait is determined by sending a `tag` message to *sender* and interpreting it as a font trait mask for a `convertFont:toNotHaveTrait:` (page 1223) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addFontTrait:` (page 1216)
- `modifyFont:` (page 1230)
- `modifyFontViaPanel:` (page 1231)

Declared In
`NSFontManager.h`

selectedFont

Returns the last font recorded.

```
- (NSFont *)selectedFont
```

Return Value
The last font recorded with a `setSelectedFont:isMultiple:` (page 1237) message

Discussion
While fonts are being converted in response to a `convertFont:` (page 1220) message, you can determine the font selected in the Font panel like this:

```
NSFontManager *fontManager = [NSFontManager sharedFontManager];
panelFont = [fontManager convertFont:[fontManager selectedFont]];
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isMultiple` (page 1230)

Related Sample Code
QTKitTimeCode

Declared In
`NSFontManager.h`

sendAction

Sends the receiver's action message up the responder chain, initiating a font change for whatever conversion and trait to change were last requested.

```
- (BOOL)sendAction
```

Return Value
YES if some object handled the changeFont: (page 1239) message; NO if the message went unheard.

Discussion
By default, the receiver's action message is changeFont: (page 1239).

This method is used internally by the font conversion system. You should never need to invoke it directly. Instead, use the action methods such as addFontTrait: (page 1216) or modifyFontViaPanel: (page 1231).

Availability
Available in Mac OS X v10.0 and later.

See Also
- setAction: (page 1235)

Declared In
NSFontManager.h

setAction:

Sets the action that's sent to the first responder, when the user selects a new font from the Font panel or chooses a command from the Font menu, to the given selector.

```
- (void)setAction:(SEL)aSelector
```

Parameters
aSelector
The selector to set.

Discussion
The default action is changeFont: (page 1239). You should rarely need to change this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- action (page 1215)

Declared In
NSFontManager.h

setDelegate:

Sets the receiver's delegate to the given object.

```
- (void)setDelegate:(id)anObject
```

Parameters
anObject
The new delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `delegate` (page 1226)

Related Sample Code
PDF Annotation Editor

Declared In
`NSFontManager.h`

setEnabled:

Controls whether the font conversion system's user interface items (the Font panel and Font menu items) are enabled.

```
- (void)setEnabled:(BOOL)flag
```

Parameters

flag
If `YES`, they're enabled; if `NO`, they're disabled.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setEnabled:` (page 1251) (NSFontPanel)
- `isEnabled` (page 1229)

Declared In
`NSFontManager.h`

setFontMenu:

Records the given menu as the application's Font menu.

```
- (void)setFontMenu:(NSMenu *)aMenu
```

Parameters

aMenu
The new Font menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `fontMenu:` (page 1227)

Declared In
`NSFontManager.h`

setSelectedAttributes:isMultiple:

Informs the paragraph and character formatting panels when text in a selection has changed attributes.

```
- (void)setSelectedAttributes:(NSDictionary *)attributes isMultiple:(BOOL)flag
```

Parameters

attributes

The new attributes.

flag

If YES, informs the panel that multiple fonts or attributes are enclosed within the selection.

Discussion

This method is used primarily by NSTextView.

Availability

Available in Mac OS X v10.3 and later.

See Also

- convertAttributes: (page 1220)

Declared In

NSFontManager.h

setSelectedFont:isMultiple:

Records the given font as the currently selected font and updates the Font panel to reflect this.

```
- (void)setSelectedFont:(NSFont *)aFont isMultiple:(BOOL)flag
```

Parameters

aFont

The font to set as selected.

flag

If YES, the Font panel indicates that more than one font is contained in the selection; if NO, it does not.

Discussion

An object that manipulates fonts should invoke this method whenever it becomes first responder and whenever its selection changes. It shouldn't invoke this method in the process of handling a changeFont: (page 1239) message, as this causes the font manager to lose the information necessary to effect the change. After all fonts have been converted, the font manager itself records the new selected font.

Availability

Available in Mac OS X v10.0 and later.

See Also

- selectedFont (page 1234)
- isMultiple (page 1230)

Related Sample Code

QTKitTimeCode

Quartz Composer WWDC 2005 TextEdit

Declared In

NSFontManager.h

setTarget:

Sets the target for the `sendAction` (page 1234) method.

```
- (void)setTarget:(id)aTarget
```

Parameters

aTarget

The target to set.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- target` (page 1238)

Related Sample Code
CoreTextArcCocoa

Declared In
`NSFontManager.h`

target

Returns the target for the `sendAction` (page 1234) method.

```
- (id)target
```

Return Value
The target for the receiver's `sendAction` (page 1234) method.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setTarget:` (page 1238)

Declared In
`NSFontManager.h`

traitsOfFont:

Returns the traits of the given font.

```
- (NSFontTraitMask)traitsOfFont:(NSFont *)aFont
```

Parameters

aFont

The font whose traits are returned.

Return Value
The font traits, returned as a mask created by combining values listed in "Constants" (page 1241) with the C bitwise OR operator.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTKitTimeCode

Declared In
`NSFontManager.h`

weightOfFont:

Returns a rough numeric measure the weight of the given font.

```
- (NSInteger)weightOfFont:(NSFont *)aFont
```

Parameters

aFont

The font whose weight is returned.

Return Value
A rough numeric measure the weight of the given font, where 0 indicates the lightest possible weight, 5 indicates a normal or book weight, and 9 or more indicates a bold or heavier weight.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFontManager.h`

Delegate Methods

changeFont:

Informs responders of a font change.

```
- (void)changeFont:(id)sender
```

Parameters

sender

The object that sent the message.

Discussion
Generally this change is because the user changed the font either in the selection of a rich text field or in a whole plain text field. Any object that contains a font the user can change must respond to the `changeFont:` message by sending a `convertFont:` (page 1220) message back to *sender* (an NSFontManager object) for each font in the selection. For more information, see "Responding to Font Changes".

Be aware that `selectedFont` (page 1234) at this point may return unpredictable results. The font returned from this method may not be the last font selected, or there may be multiple fonts selected at the time `changeFont:` is called. The use of `selectedFont` from within `changeFont:` is strongly discouraged.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addFontTrait:` (page 1216)
- `convertFont:toHaveTrait:` (page 1222)
- `convertFont:toFace:` (page 1221)
- `convertFont:toFamily:` (page 1222)
- `convertFont:toNotHaveTrait:` (page 1223)
- `convertFont:toSize:` (page 1224)
- `convertWeight:ofFont:` (page 1225)
- `convertFont:` (page 1220)
- `removeFontTrait:` (page 1233)
- `modifyFontViaPanel:` (page 1231)
- `modifyFont:` (page 1230)

Declared In
`NSFontManager.h`

fontManager:willIncludeFont:

Requests permission from the Font panel delegate to display the given font name in the Font panel.

```
- (BOOL)fontManager:(id)theFontManager willIncludeFont:(NSString *)fontName
```

Parameters

`theFontManager`
The font manager making the request.

`fontName`
The full PostScript name of the font to display, such as Helvetica-BoldOblique or Helvetica-Narrow-Bold.

Return Value
If the Font panel delegate returns `YES`, *`fontName`* is listed; if the delegate returns `NO`, it isn't.

Discussion
In Mac OS X versions 10.2 and earlier, this method is invoked repeatedly as necessary whenever the Font panel needs updating, such as when the Font panel is first loaded, and when the user selects a family name to see which typefaces in that family are available. Your implementation should execute fairly quickly to ensure the responsiveness of the Font panel.

Important: This delegate method is not called in Mac OS X versions 10.3 and later.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFontManager.h`

Constants

Font Collection Mask

This constant specifies options accepted by `addCollection:options:` (page 1216).

```
enum {
   NSFontCollectionApplicationOnlyMask = 1 << 0
};
```

Constants

`NSFontCollectionApplicationOnlyMask`

Makes the collection available only to the application. This option is not yet implemented.

Available in Mac OS X v10.3 and later.

Declared in `NSFontManager.h`.

Declared In

`NSFontManager.h`

NSFontTraitMask

Mask of traits assigned to a font, assigned using the values in `Font traits` (page 1241).

```
typedef unsigned int NSFontTraitMask;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSFontManager.h`

Font traits

Font traits defined and supported by `NSFontManager`.

```
enum {
   NSItalicFontMask = 0x00000001,
   NSBoldFontMask = 0x00000002,
   NSUnboldFontMask = 0x00000004,
   NSNonStandardCharacterSetFontMask = 0x00000008,
   NSNarrowFontMask = 0x00000010,
   NSExpandedFontMask = 0x00000020,
   NSCondensedFontMask = 0x00000040,
   NSSmallCapsFontMask = 0x00000080,
   NSPosterFontMask = 0x00000100,
   NSCompressedFontMask = 0x00000200,
   NSFixedPitchFontMask = 0x00000400,
   NSUnitalicFontMask = 0x01000000
};
```

Constants

`NSItalicFontMask`

A mask that specifies an italic font.

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSBoldFontMask`

A mask that specifies a bold font.

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSUnboldFontMask`

A mask that specifies a font that is not bold.

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSNonStandardCharacterSetFontMask`

A mask that specifies a font that uses a non-standard character set.

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSNarrowFontMask`

A mask that specifies a narrow font.

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSExpandedFontMask`

A mask that specifies an expanded font.

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSCondensedFontMask`

A mask that specifies a condensed font.

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSSmallCapsFontMask`
A mask that specifies a small-caps font.
Available in Mac OS X v10.0 and later.
Declared in `NSFontManager.h`.

`NSPosterFontMask`
A mask that specifies a poster-style font.
Available in Mac OS X v10.0 and later.
Declared in `NSFontManager.h`.

`NSCompressedFontMask`
A mask that specifies a compressed font.
Available in Mac OS X v10.0 and later.
Declared in `NSFontManager.h`.

`NSFixedPitchFontMask`
A mask that specifies a fixed pitch font.
Available in Mac OS X v10.0 and later.
Declared in `NSFontManager.h`.

`NSUnitalicFontMask`
A mask that specifies a font that is not italic.
Available in Mac OS X v10.0 and later.
Declared in `NSFontManager.h`.

Discussion
`NSFontManager` categorizes fonts according to a small set of traits. You can convert fonts by adding and removing individual traits, and you can get a font with a specific combination of traits.

These pairs of traits are mutually exclusive:

`NSCondensedFontMask` and `NSExpandedFontMask`
`NSBoldFontMask` and `NSUnboldFontMask`
`NSItalicFontMask` and `NSUnitalicFontMask`

Declared In
`NSFontManager.h`

NSFontAction

These constants specify what action `modifyFont:` (page 1230) will take.

```
typedef enum _NSFontAction {
    NSNoFontChangeAction = 0,
    NSViaPanelFontAction = 1,
    NSAddTraitFontAction = 2,
    NSSizeUpFontAction = 3,
    NSSizeDownFontAction = 4,
    NSHeavierFontAction = 5,
    NSLighterFontAction = 6,
    NSRemoveTraitFontAction = 7
} NSFontAction;
```

Constants

`NSNoFontChangeAction`
No action; the font is returned unchanged.

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSViaPanelFontAction`
Converts the font according to the `NSFontPanel` method `panelConvertFont:`.

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSAddTraitFontAction`
Converts the font to have an additional trait using `convertFont:toHaveTrait:` (page 1222).

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSSizeUpFontAction`
Converts the font to a larger size using `convertFont:toSize:` (page 1224).

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSSizeDownFontAction`
Converts the font to a smaller size using `convertFont:toSize:` (page 1224).

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSHeavierFontAction`
Converts the font to a heavier weight using `convertWeight:ofFont:` (page 1225).

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSLighterFontAction`
Converts the font to a lighter weight using `convertWeight:ofFont:` (page 1225).

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

`NSRemoveTraitFontAction`
Converts the font to remove a trait using `convertFont:toNotHaveTrait:` (page 1223).

Available in Mac OS X v10.0 and later.

Declared in `NSFontManager.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFontManager.h`

NSFontPanel Class Reference

Inherits from	NSPanel : NSWindow : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSWindow) NSAnimatablePropertyContainer (NSWindow) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSFontPanel.h
Companion guide	Font Panel
Related sample code	Aperture Edit Plugin - Borders & Titles CoreAnimationText CoreTextArcCocoa PDF Annotation Editor QTKitTimeCode

Overview

The `NSFontPanel` class implements the Font panel—a user interface object that displays a list of available fonts, letting the user preview them and change the font used to display text. The actual changes are made through conversion messages sent to the shared `NSFontManager` instance. There's only one Font panel for each application.

Tasks

Getting the Font Panel

`+ sharedFontPanel` (page 1248)
Returns the single `NSFontPanel` instance for the application, creating it if necessary.

`+ sharedFontPanelExists` (page 1249)
Returns `YES` if the shared Font panel has been created, `NO` if it hasn't.

Enabling Font Changes

- `isEnabled` (page 1250)
  Indicates whether the receiver's Set button is enabled.
- `reloadDefaultFontFamilies` (page 1250)
  Triggers a reload to the default state, so that the delegate is called.
- `setEnabled:` (page 1251) Available in Mac OS X v10.0 through Mac OS X v10.5
  Specifies whether the receiver's Set button is enabled.

Updating the Font Panel

- `setPanelFont:isMultiple:` (page 1252)
  Sets the selected font in the receiver to the specified font.

Converting Fonts

- `panelConvertFont:` (page 1250)
  Converts the specified font using the settings in the receiver, with the aid of the shared `NSFontManager` if necessary.

Working in Modal Loops

- `worksWhenModal` (page 1252)
  Indicates whether the receiver allows fonts to be changed in modal windows and panels.

Setting an Accessory View

- `setAccessoryView:` (page 1251)
  Establishes the specified view as the receiver's accessory view, allowing you to add custom controls to your application's Font panel without having to create a subclass.
- `accessoryView` (page 1249)
  Returns the receiver's accessory view.

Class Methods

sharedFontPanel

Returns the single `NSFontPanel` instance for the application, creating it if necessary.

```
+ (NSFontPanel *)sharedFontPanel
```

Return Value
The `NSFontPanel` instance for the application.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ sharedFontPanelExists` (page 1249)
`+ setFontPanelFactory:` (page 1214) (`NSFontManager`)

Related Sample Code
Aperture Edit Plugin - Borders & Titles
CoreAnimationText
CoreTextArcCocoa
PDF Annotation Editor

Declared In
`NSFontPanel.h`

sharedFontPanelExists

Returns `YES` if the shared Font panel has been created, `NO` if it hasn't.

```
+ (BOOL)sharedFontPanelExists
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ sharedFontPanel` (page 1248)

Declared In
`NSFontPanel.h`

Instance Methods

accessoryView

Returns the receiver's accessory view.

```
- (NSView *)accessoryView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAccessoryView:` (page 1251)

Declared In
`NSFontPanel.h`

isEnabled

Indicates whether the receiver's Set button is enabled.

```
- (BOOL)isEnabled
```

Return Value
`YES` if the receiver's Set button is enabled; `NO` if it isn't.

Discussion
The receiver continues to reflect the font of the selection for cooperating text objects regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setEnabled:` (page 1251)

Declared In
`NSFontPanel.h`

panelConvertFont:

Converts the specified font using the settings in the receiver, with the aid of the shared `NSFontManager` if necessary.

```
- (NSFont *)panelConvertFont:(NSFont *)aFont
```

Parameters

aFont
The font to be converted.

Return Value
The converted font, or *aFont* itself if it can't be converted.

Discussion
For example, if *aFont* is Helvetica Oblique 12.0 point and the user has selected the Times font family (and nothing else) in the Font panel, the font returned is Times Italic 12.0 point.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- convertFont:` (page 1220) (NSFontManager)

Declared In
`NSFontPanel.h`

reloadDefaultFontFamilies

Triggers a reload to the default state, so that the delegate is called.

```
- (void)reloadDefaultFontFamilies
```

Discussion
This reloading provides the delegate opportunity to scrutinize the default list of fonts to be displayed in the panel.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSFontPanel.h

setAccessoryView:

Establishes the specified view as the receiver's accessory view, allowing you to add custom controls to your application's Font panel without having to create a subclass.

```
- (void)setAccessoryView:(NSView *)aView
```

Parameters

aView

The view to set.

Availability
Available in Mac OS X v10.0 and later.

See Also
- accessoryView (page 1249)

Declared In
NSFontPanel.h

setEnabled:

Specifies whether the receiver's Set button is enabled.

```
- (void)setEnabled:(BOOL)flag
```

Parameters

flag

If YES the Set button is enabled; if NO, it's disabled.

Discussion
The receiver continues to reflect the font of the selection for cooperating text objects regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isEnabled (page 1250)

Declared In
NSFontPanel.h

setPanelFont:isMultiple:

Sets the selected font in the receiver to the specified font.

```
- (void)setPanelFont:(NSFont *)aFont isMultiple:(BOOL)flag
```

Parameters

aFont

The font to be selected.

flag

If NO, selects the specified font; otherwise selects no font and displays a message in the preview area indicating that multiple fonts are selected.

Discussion

You normally don't use this method directly; instead, you send `setSelectedFont:isMultiple:` (page 1237) to the shared `NSFontManager`, which in turn invokes this method.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Aperture Edit Plugin - Borders & Titles

CoreTextArcCocoa

PDF Annotation Editor

Declared In

`NSFontPanel.h`

worksWhenModal

Indicates whether the receiver allows fonts to be changed in modal windows and panels.

```
- (BOOL)worksWhenModal
```

Return Value

YES, regardless of the setting established using the `NSPanel` method `setWorksWhenModal:` (page 1862).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `worksWhenModal` (page 3409) (`NSWindow`)
- `worksWhenModal` (page 1862) (`NSPanel`)

Declared In

`NSFontPanel.h`

Constants

Tags of Views in the FontPanel

These constants are obsolete and should not be used. (Deprecated. Use the method and constants described in *NSFontPanelValidation Protocol Reference* instead.)

```
enum {
    NSFPPreviewButton = 131,
    NSFPRevertButton = 130,
    NSFPSetButton = 132,
    NSFPPreviewField = 128,
    NSFPSizeField = 129,
    NSFPSizeTitle = 133,
    NSFPCurrentField = 134
};
```

Constants

`NSFPPreviewButton`
Show the Preview button.
Available in Mac OS X v10.0 and later.
Declared in `NSFontPanel.h`.

`NSFPRevertButton`
Show the Revert button.
Available in Mac OS X v10.0 and later.
Declared in `NSFontPanel.h`.

`NSFPSetButton`
Show the Set button.
Available in Mac OS X v10.0 and later.
Declared in `NSFontPanel.h`.

`NSFPPreviewField`
Show the Preview field.
Available in Mac OS X v10.0 and later.
Declared in `NSFontPanel.h`.

`NSFPSizeField`
Show the Size field.
Available in Mac OS X v10.0 and later.
Declared in `NSFontPanel.h`.

`NSFPSizeTitle`
Show the Size title.
Available in Mac OS X v10.0 and later.
Declared in `NSFontPanel.h`.

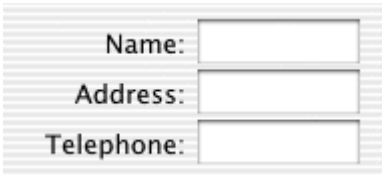
`NSFPCurrentField`
Show the Current field.
Available in Mac OS X v10.0 and later.
Declared in `NSFontPanel.h`.

NSForm Class Reference

Inherits from	NSMatrix : NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSMatrix) NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSForm.h
Companion guides	Forms Matrix Programming Guide
Related sample code	SimpleComboBox SourceView

Overview

An `NSForm` object is a vertical matrix of `NSFormCell` objects. Here's an example:



An `NSForm` object uses NSFormCell to implement its user interface.

Tasks

Adding and Removing Entries

`- addEntry:` (page 1257)
Adds a new entry to the end of the receiver and gives it the specified title.

- `insertEntry:atIndex:` (page 1259)
  Inserts an entry with the specified title into the receiver.
- `removeEntryAtIndex:` (page 1259)
  Removes and releases the entry at the specified index.

Changing the Appearance of All the Entries

- `setBezeled:` (page 1260)
  Sets whether the receiver's entries should display a bezel around their editable text.
- `setBordered:` (page 1260)
  Sets whether the receiver's entries should display a border around their editable text fields.
- `setEntryWidth:` (page 1261)
  Sets the width of all the entries in the receiver.
- `setFrameSize:` (page 1261)
  Sets the size of the receiver's frame size to the specified value.
- `setInterlineSpacing:` (page 1262)
  Sets the spacing between entries
- `setTitleAlignment:` (page 1263)
  Sets the alignment for all of the entry titles.
- `setTitleBaseWritingDirection:` (page 1263)
  Sets the writing direction for the title of every control embedded in the form.
- `setTextAlignment:` (page 1262)
  Sets the alignment for all of the receiver's editable text.
- `setTextBaseWritingDirection:` (page 1262)
  Sets the writing direction for the text content of every control embedded in the form.
- `setTitleFont:` (page 1264)
  Sets the font for all of the entry titles.
- `setTextFont:` (page 1263)
  Sets the font for all of the receiver's editable text fields

Getting Cells and Indices

- `indexOfCellWithTag:` (page 1258)
  Returns the index of the entry whose tag is *`tag`*.
- `indexOfSelectedItem` (page 1259)
  Returns the index of the selected entry.
- `cellAtIndex:` (page 1257)
  Returns the entry at the specified index.

Displaying a Cell

- `drawCellAtIndex:` (page 1258)
  Displays the entry at the specified index.

Editing Text

`- selectTextAtIndex:` (page 1260)
Selects the entry at the specified index.

Instance Methods

addEntry:

Adds a new entry to the end of the receiver and gives it the specified title.

```
- (NSFormCell *)addEntry:(NSString *)title
```

Parameters

title
The title for the new form entry.

Return Value
The form cell object that was created for the entry.

Discussion
The new entry has no tag, target, or action, but is enabled and editable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `insertEntry:atIndex:` (page 1259)
- `setEditable:` (page 584) (NSCell)
- `setTag:` (page 69) (NSActionCell)
- `setTarget:` (page 70) (NSActionCell)
- `setAction:` (page 65) (NSActionCell)
- `setEnabled:` (page 67) (NSActionCell)

Declared In
`NSForm.h`

cellAtIndex:

Returns the entry at the specified index.

```
- (id)cellAtIndex:(NSInteger)entryIndex
```

Parameters

entryIndex
The index of the desired entry.

Return Value
The form cell object at the specified index.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfCellWithTag:` (page 1258)
- `indexOfSelectedItem` (page 1259)

Declared In
`NSForm.h`

drawCellAtIndex:

Displays the entry at the specified index.

```
- (void)drawCellAtIndex:(NSInteger)entryIndex
```

Parameters

entryIndex
The index of the entry to draw.

Discussion
Because this method is called automatically whenever a cell needs drawing, you never need to invoke it explicitly. It is included in the API so you can override it if you subclass `NSFormCell`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfCellWithTag:` (page 1258)
- `indexOfSelectedItem` (page 1259)

Declared In
`NSForm.h`

indexOfCellWithTag:

Returns the index of the entry whose tag is *tag*.

```
- (NSInteger)indexOfCellWithTag:(NSInteger)tag
```

Parameters

tag
The tag of the desired entry.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `tag` (page 605) (NSCell)

Declared In
`NSForm.h`

indexOfSelectedItem

Returns the index of the selected entry.

```
- (NSInteger)indexOfSelectedItem
```

Return Value
The index of the selected entry, or `-1` if no entry is selected.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSForm.h`

insertEntry:atIndex:

Inserts an entry with the specified title into the receiver.

```
- (NSFormCell *)insertEntry:(NSString *)title atIndex:(NSInteger)entryIndex
```

Parameters
title
The title for the new form entry.

entryIndex
The zero-based index at which to insert the entry.

Return Value
The form cell object that was created for the entry.

Discussion
The new entry has no tag, target, or action, but is enabled and editable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addEntry:` (page 1257)
- `removeEntryAtIndex:` (page 1259)

Declared In
`NSForm.h`

removeEntryAtIndex:

Removes and releases the entry at the specified index.

```
- (void)removeEntryAtIndex:(NSInteger)entryIndex
```

Parameters
entryIndex
The zero-based index identifying the desired entry.

Discussion
If the specified index is invalid, this method does nothing.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSForm.h

selectTextAtIndex:

Selects the entry at the specified index.

```
- (void)selectTextAtIndex:(NSInteger)entryIndex
```

Parameters

entryIndex
The index of the entry to select. If the specified index is invalid, this method does nothing.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSForm.h

setBezeled:

Sets whether the receiver's entries should display a bezel around their editable text.

```
- (void)setBezeled:(BOOL)flag
```

Parameters

flag
YES to display a bezel around all entries; otherwise, NO to show no bezel around all entries.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setBordered: (page 1260)
- isBezeled (page 566) (NSCell)

Declared In
NSForm.h

setBordered:

Sets whether the receiver's entries should display a border around their editable text fields.

```
- (void)setBordered:(BOOL)flag
```

Parameters

flag

YES to display a border around all entries; otherwise, NO to show no border around all entries.

Discussion

The border is drawn as a thin line around the editable text field. An entry can have a border or a bezel, but not both.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setBezeled: (page 1260)
- isBordered (page 566) (NSCell)

Declared In

NSForm.h

setEntryWidth:

Sets the width of all the entries in the receiver.

```
- (void)setEntryWidth:(CGFloat)width
```

Parameters

width

The width of all entries, measured in points in the user coordinate space. This value represents the width of both the title and the text field.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSForm.h

setFrameSize:

Sets the size of the receiver's frame size to the specified value.

```
- (void)setFrameSize:(NSSize)newSize
```

Parameters

newSize

The new size of the form.

Discussion

The width of NSFormCell objects always match the width of their encompassing NSForm object. The cell width is always changed to match the view regardless of the value returned by autosizesCells (page 1556).

Availability

Available in Mac OS X v10.3 and later.

Declared In

NSForm.h

setInterlineSpacing:

Sets the spacing between entries

```
- (void)setInterlineSpacing:(CGFloat)spacing
```

Parameters

spacing

The spacing between entries, measured in points in the user coordinate space.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSForm.h

setTextAlignment:

Sets the alignment for all of the receiver's editable text.

```
- (void)setTextAlignment:(NSInteger)alignment
```

Parameters

alignment

The alignment can be one of the following constants: `NSRightTextAlignment`, `NSCenterTextAlignment`, or `NSLeftTextAlignment`.

Discussion
The default alignment is `NSLeftTextAlignment`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitleAlignment:` (page 1263)

Declared In
NSForm.h

setTextBaseWritingDirection:

Sets the writing direction for the text content of every control embedded in the form.

```
- (void)setTextBaseWritingDirection:(NSWritingDirection)writingDirection
```

Parameters

writingDirection

This value can be one of the following constants: `NSWritingDirectionNatural`, `NSWritingDirectionLeftToRight`, or `NSWritingDirectionRightToLeft`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `baseWritingDirection` (page 815) (NSControl)

Declared In
`NSForm.h`

setTextFont:

Sets the font for all of the receiver's editable text fields

```
- (void)setTextFont:(NSFont *)font
```

Parameters

font

The font to use for all editable text fields.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTextFont:` (page 1263)

Declared In
`NSForm.h`

setTitleAlignment:

Sets the alignment for all of the entry titles.

```
- (void)setTitleAlignment:(NSTextAlignment)alignment
```

Parameters

alignment

The alignment can be one of the following constants: `NSRightTextAlignment`, `NSCenterTextAlignment`, or `NSLeftTextAlignment`.

Discussion
The default alignment is `NSRightTextAlignment`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTextAlignment:` (page 1262)

Declared In
`NSForm.h`

setTitleBaseWritingDirection:

Sets the writing direction for the title of every control embedded in the form.

```
- (void)setTitleBaseWritingDirection:(NSWritingDirection)writingDirection
```

Parameters

writingDirection

This value can be one of the following constants: `NSWritingDirectionNatural`, `NSWritingDirectionLeftToRight`, or `NSWritingDirectionRightToLeft`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSForm.h`

setTitleFont:

Sets the font for all of the entry titles.

```
- (void)setTitleFont:(NSFont *)font
```

Parameters

font

The font to use for all entry titles.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTextFont:` (page 1263)

Declared In
`NSForm.h`

NSFormCell Class Reference

Inherits from	NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSFormCell.h
Companion guide	Forms
Related sample code	DragNDropOutlineView SimpleComboBox TemperatureConverter

Overview

The `NSFormCell` class is used to implement text entry fields in a form. The left part of an `NSFormCell` object contains a title. The right part contains an editable text entry field.

An `NSFormCell` object implements the user interface of an NSForm object.

Tasks

Initializing an NSFormCell

`- initTextCell:` (page 1267)
Returns an `NSFormCell` object initialized with the specified title string.

Asking About a Cell's Appearance

`- isOpaque` (page 1268)
Returns a Boolean value indicating whether the title is empty and an opaque bezel is set.

Asking About a Cell's Title

- `attributedTitle` (page 1267)
  Returns the title of the cell as an attributed string.
- `title` (page 1272)
  Returns the receiver's title.
- `titleAlignment` (page 1272)
  Returns the alignment of the title.
- `titleBaseWritingDirection` (page 1273)
  Returns the default writing direction used to render the form cell's title.
- `titleFont` (page 1273)
  Returns the font used to draw the receiver's title.
- `titleWidth` (page 1273)
  Returns the width of the title field.
- `titleWidth:` (page 1274)
  Returns the width of the title field constrained to the specified size.

Changing the Cell's Title

- `setAttributedTitle:` (page 1269)
  Sets the receiver's title using an attributed string.
- `setTitle:` (page 1270)
  Sets the receiver's title to the specified plain-text string.
- `setTitleAlignment:` (page 1270)
  Sets the alignment of the title.
- `setTitleBaseWritingDirection:` (page 1271)
  Sets the default writing direction used to render the form cell's title.
- `setTitleFont:` (page 1271)
  Sets the font of the receiver's title.
- `setTitleWidth:` (page 1271)
  Sets the width of the title.

Setting a Keyboard Equivalent

- `setTitleWithMnemonic:` (page 1272)
  Sets the cell title and mnemonic character.

Asking About Placeholder Values

- `placeholderAttributedString` (page 1268)
  Returns the cell's attributed placeholder string.
- `placeholderString` (page 1268)
  Returns the cell's plain text placeholder string.

- `setPlaceholderAttributedString:` (page 1269)
  Sets the attributed placeholder string for the cell.
- `setPlaceholderString:` (page 1269)
  Sets the plain-text placeholder string for the cell.

Instance Methods

attributedTitle

Returns the title of the cell as an attributed string.

```
- (NSAttributedString *)attributedTitle
```

Return Value
The title of the cell.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFormCell.h`

initTextCell:

Returns an `NSFormCell` object initialized with the specified title string.

```
- (id)initTextCell:(NSString *)aString
```

Parameters

aString
The title for the new form cell object.

Return Value
An initialized `NSFormCell` object.

Discussion
The contents of the cell's editable text entry field are set to the empty string (@""). The font for both title and text is the user's chosen system font in 12.0 point, and the text area is drawn with a bezel. This method is the designated initializer for the `NSFormCell` class.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTitle:` (page 1270)

Declared In
`NSFormCell.h`

isOpaque

Returns a Boolean value indicating whether the title is empty and an opaque bezel is set.

```
- (BOOL)isOpaque
```

Return Value
`YES` if the title is empty and an opaque bezel is set; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFormCell.h`

placeholderAttributedString

Returns the cell's attributed placeholder string.

```
- (NSAttributedString *)placeholderAttributedString
```

Return Value
The attributed placeholder string, or `nil` if the cell has no attributed placeholder string.

Discussion
If this method returns `nil`, you can also call `placeholderString` to see if the cell has a plain text placeholder string.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `placeholderString` (page 1268)
- `setPlaceholderAttributedString:` (page 1269)

Declared In
`NSFormCell.h`

placeholderString

Returns the cell's plain text placeholder string.

```
- (NSString *)placeholderString
```

Return Value
The plain-text placeholder string, or `nil` if the cell has no plain-text placeholder string.

Discussion
If this method returns `nil`, you can also call `placeholderAttributedString` to see if the cell has an attributed placeholder string.

Availability
Available in Mac OS X v10.4 and later.

See Also
- placeholderAttributedString (page 1268)
- setPlaceholderString: (page 1269)

Declared In
NSFormCell.h

setAttributedTitle:

Sets the receiver's title using an attributed string.

```
- (void)setAttributedTitle:(NSAttributedString *)anAttributedString
```

Parameters
anAttributedString
The formatted title of the cell.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSFormCell.h

setPlaceholderAttributedString:

Sets the attributed placeholder string for the cell.

```
- (void)setPlaceholderAttributedString:(NSAttributedString *)string
```

Parameters
string
The attributed placeholder string.

Discussion
Note that invoking this method clears out any plain text string set by calling the setPlaceholderString: (page 1269) method.

Availability
Available in Mac OS X v10.4 and later.

See Also
- placeholderAttributedString (page 1268)
- setPlaceholderString: (page 1269)

Declared In
NSFormCell.h

setPlaceholderString:

Sets the plain-text placeholder string for the cell.

```
- (void)setPlaceholderString:(NSString *)string
```

Parameters

string

The plain-text placeholder string.

Discussion

Note that invoking this method clears out any attributed string set by the `setPlaceholderAttributedString:` (page 1269) method.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- placeholderString` (page 1268)

`- setPlaceholderAttributedString:` (page 1269)

Declared In

`NSFormCell.h`

setTitle:

Sets the receiver's title to the specified plain-text string.

```
- (void)setTitle:(NSString *)aString
```

Parameters

aString

The plain-text title of the cell.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSFormCell.h`

setTitleAlignment:

Sets the alignment of the title.

```
- (void)setTitleAlignment:(NSTextAlignment)alignment
```

Parameters

alignment

The alignment can be one of the following constants: `NSRightTextAlignment`, `NSCenterTextAlignment`, or `NSLeftTextAlignment`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSFormCell.h`

setTitleBaseWritingDirection:

Sets the default writing direction used to render the form cell's title.

```
- (void)setTitleBaseWritingDirection:(NSWritingDirection)writingDirection
```

Parameters

writingDirection

This value can be one of the following constants: `NSWritingDirectionNatural`, `NSWritingDirectionLeftToRight`, or `NSWritingDirectionRightToLeft`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTitleBaseWritingDirection:` (page 1271)
- `baseWritingDirection` (page 547) (NSCell)

Declared In
`NSFormCell.h`

setTitleFont:

Sets the font of the receiver's title.

```
- (void)setTitleFont:(NSFont *)font
```

Parameters

font

The font to use.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFormCell.h`

setTitleWidth:

Sets the width of the title.

```
- (void)setTitleWidth:(CGFloat)width
```

Parameters

width

The width of the title, measured in points in the user coordinate space.

Discussion
You usually do not need to invoke this method. The Application Kit automatically sets the title width whenever the title changes. If the automatic width doesn't suit your needs, though, you can use this method to set the width explicitly.

Once you have set the width this way, the Application Kit stops setting the width automatically; you must invoke this method every time the title changes. If you want the Application Kit to resume automatic width assignments, invoke this method with a negative *width* value.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSFormCell.h

setTitleWithMnemonic:

Sets the cell title and mnemonic character.

```
- (void)setTitleWithMnemonic:(NSString *)titleWithAmpersand
```

Parameters

titleWithAmpersand
The title of the cell, including a mnemonic identifier. To specify the mnemonic character, place an ampersand (&) in the front of the desired character.

Discussion
Mnemonics are not supported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTitle: (page 1270)

Declared In
NSFormCell.h

title

Returns the receiver's title.

```
- (NSString *)title
```

Return Value
The title of the cell. The default value is "Field:".

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSFormCell.h

titleAlignment

Returns the alignment of the title.

```
- (NSTextAlignment)titleAlignment
```

Return Value
The alignment can be one of the following values: `NSLeftTextAlignment`, `NSCenterTextAlignment`, or `NSRightTextAlignment`. The default alignment is `NSRightTextAlignment`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFormCell.h`

titleBaseWritingDirection

Returns the default writing direction used to render the form cell's title.

```
- (NSWritingDirection)titleBaseWritingDirection
```

Return Value
One of the following constants: `NSWritingDirectionNatural`, `NSWritingDirectionLeftToRight`, or `NSWritingDirectionRightToLeft`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setTitleBaseWritingDirection:` (page 1271)

Declared In
`NSFormCell.h`

titleFont

Returns the font used to draw the receiver's title.

```
- (NSFont *)titleFont
```

Return Value
The font object used for the title.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFormCell.h`

titleWidth

Returns the width of the title field.

```
- (CGFloat)titleWidth
```

Return Value
The width of the title field, measured in points in the user coordinate space.

Discussion
If you set the width using `setTitleWidth:` (page 1271), this method returns the value you set; otherwise, it returns the width calculated automatically by the Application Kit.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- titleWidth:` (page 1274)

Declared In
`NSFormCell.h`

titleWidth:

Returns the width of the title field constrained to the specified size.

```
- (CGFloat)titleWidth:(NSSize)aSize
```

Parameters

aSize

The maximum size of the field when calculated by the Application Kit.

Return Value
The width of the title field, measured in points in the user coordinate space.

Discussion
If you set the width using `setTitleWidth:` (page 1271), this method returns the value you set; otherwise, the Application Kit calculates the width, constraining the field size to the specified value.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- titleWidth` (page 1273)

Declared In
`NSFormCell.h`

NSGlyphGenerator Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSGlyphGenerator.h
Availability	Available in Mac OS X v10.3 and later.
Companion guides	Text System Overview Text Layout Programming Guide

Overview

An `NSGlyphGenerator` object performs the initial, nominal glyph generation phase in the layout process.

The nominal glyph generation pass essentially generates one glyph per character; the typesetter may later make substitutions in the glyph stream, for example, changing an acute accent glyph followed by an "e" glyph into a single acute-accented "é" glyph.

`NSGlyphGenerator` communicates via the `NSGlyphStorage` protocol. An example of a class conforming to the protocol is `NSLayoutManager`.

Tasks

Obtaining a Glyph Generator

`+ sharedGlyphGenerator` (page 1276)
Returns a shared instance of `NSGlyphGenerator`.

Generating Glyphs

`- generateGlyphsForGlyphStorage:desiredNumberOfCharacters:glyphIndex:characterIndex:` (page 1276)
Generates glyphs for the specified glyph storage object (`NSLayoutManager` by default).

Class Methods

sharedGlyphGenerator

Returns a shared instance of `NSGlyphGenerator`.

```
+ (id)sharedGlyphGenerator
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSGlyphGenerator.h`

Instance Methods

generateGlyphsForGlyphStorage:desiredNumberOfCharacters:glyphIndex:characterIndex:

Generates glyphs for the specified glyph storage object (`NSLayoutManager` by default).

```
- (void)generateGlyphsForGlyphStorage:(id < NSGlyphStorage >)glyphStorage
    desiredNumberOfCharacters:(NSUInteger)nChars glyphIndex:(NSUInteger *)glyphIndex
     characterIndex:(NSUInteger *)charIndex
```

Discussion
Generates glyphs for the glyph storage object specified by *glyphStorage*, beginning with the character at *charIndex* and continuing for *nChars* characters. The *glyphIndex* specifies the index of the first glyph generated.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSGlyphGenerator.h`

NSGlyphInfo Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSGlyphInfo.h
Availability	Available in Mac OS X v10.2 and later.
Companion guide	Font Handling

Overview

An `NSGlyphInfo` object represents a glyph attribute value (`NSGlyphInfoAttributeName`) in an attributed string. `NSGlyphInfo` allows you to override a font's specified mapping from Unicode to the glyph ID. Overriding the mapping allows you to specify a variant glyph for a given character if the font contains multiple variations for that character or to specify a glyph that doesn't have a standard mapping (such as some ligature glyphs).

Adopted Protocols

NSCoding

- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Creating an NSGlyphInfo Object

`+ glyphInfoWithCharacterIdentifier:collection:baseString:` (page 1278)
Instantiates and returns an `NSGlyphInfo` object using a character identifier and a character collection.

+ glyphInfoWithGlyph:forFont:baseString: (page 1278)
Instantiates and returns an NSGlyphInfo object using a glyph index and a specified font.

+ glyphInfoWithGlyphName:forFont:baseString: (page 1279)
Instantiates and returns an NSGlyphInfo object using a glyph name and a specified font.

Getting Information About an NSGlyphInfo Object

- characterIdentifier (page 1280)
Returns the receiver's character identifier (CID).

- characterCollection (page 1279)
Returns an NSCharacterCollection value specifying the glyph–to–character identifier mapping of the receiver.

- glyphName (page 1280)
Returns the receiver's glyph name.

Class Methods

glyphInfoWithCharacterIdentifier:collection:baseString:

Instantiates and returns an NSGlyphInfo object using a character identifier and a character collection.

```
+ (NSGlyphInfo *)glyphInfoWithCharacterIdentifier:(NSUInteger)cid
    collection:(NSCharacterCollection)characterCollection baseString:(NSString
    *)theString
```

Parameters

cid
A character identifier.

characterCollection
A string constant representing a character collection. Possible values for *characterCollection* are described in "Constants" (page 1281).

theString
The part of the attributed string the returned instance is intended to override.

Return Value
The created NSGlyphInfo object or nil if the object couldn't be created.

Availability
Available in Mac OS X v10.2 and later.

Declared In
NSGlyphInfo.h

glyphInfoWithGlyph:forFont:baseString:

Instantiates and returns an NSGlyphInfo object using a glyph index and a specified font.

```
+ (NSGlyphInfo *)glyphInfoWithGlyph:(NSGlyph)glyph forFont:(NSFont *)font
    baseString:(NSString *)theString
```

Parameters

glyph

The identifier of the glyph.

font

The font object to be associated with the returned `NSGlyphInfo` object,

theString

The part of the attributed string the returned instance is intended to override.

Return Value

The created `NSGlyphInfo` object or `nil` if the object couldn't be created.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSGlyphInfo.h`

glyphInfoWithGlyphName:forFont:baseString:

Instantiates and returns an `NSGlyphInfo` object using a glyph name and a specified font.

```
+ (NSGlyphInfo *)glyphInfoWithGlyphName:(NSString *)glyphName forFont:(NSFont *)font
     baseString:(NSString *)theString
```

Parameters

glyphName

The name of the glyph.

font

The font object to be associated with the returned `NSGlyphInfo` object,

theString

The part of the attributed string the returned instance is intended to override.

Return Value

The created `NSGlyphInfo` object or `nil` if the object couldn't be created.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSGlyphInfo.h`

Instance Methods

characterCollection

Returns an `NSCharacterCollection` value specifying the glyph–to–character identifier mapping of the receiver.

```
- (NSCharacterCollection)characterCollection
```

Discussion
This method returns `NSIdentityMappingCharacterCollection` if the receiver was instantiated with either an `NSGlyph` identifier or a glyph name. It returns other possible values if the receiver was instantiated using `glyphInfoWithCharacterIdentifier:collection:baseString:` (page 1278). These constants are described in `NSCharacterCollection` (page 1281).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSGlyphInfo.h`

characterIdentifier

Returns the receiver's character identifier (CID).

```
- (NSUInteger)characterIdentifier
```

Discussion
If the receiver was instantiated with a method other than `glyphInfoWithCharacterIdentifier:collection:baseString:` (page 1278), this method returns `NULL`.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSGlyphInfo.h`

glyphName

Returns the receiver's glyph name.

```
- (NSString *)glyphName
```

Discussion
If the receiver was instantiated with a method other than `glyphInfoWithGlyphName:forFont:baseString:` (page 1279), this method returns `nil`.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSGlyphInfo.h`

Constants

NSCharacterCollection

The following values specify the mapping of character identifiers to glyphs, and are returned by `characterCollection` (page 1279).

```
typedef enum {
    NSIdentityMappingCharacterCollection = 0,
    NSAdobeCNS1CharacterCollection       = 1,
    NSAdobeGB1CharacterCollection        = 2,
    NSAdobeJapan1CharacterCollection     = 3,
    NSAdobeJapan2CharacterCollection     = 4,
    NSAdobeKorea1CharacterCollection     = 5,
} NSCharacterCollection;
```

Constants

`NSIdentityMappingCharacterCollection`
Indicates that the character identifier is equal to the glyph index.

Available in Mac OS X v10.2 and later.

Declared in `NSGlyphInfo.h`.

`NSAdobeCNS1CharacterCollection`
Indicates the Adobe-CNS1 mapping.

Available in Mac OS X v10.2 and later.

Declared in `NSGlyphInfo.h`.

`NSAdobeGB1CharacterCollection`
Indicates the Adobe-GB1 mapping.

Available in Mac OS X v10.2 and later.

Declared in `NSGlyphInfo.h`.

`NSAdobeJapan1CharacterCollection`
Indicates the Adobe-Japan1 mapping.

Available in Mac OS X v10.2 and later.

Declared in `NSGlyphInfo.h`.

`NSAdobeJapan2CharacterCollection`
Indicates the Adobe-Japan2 mapping.

Available in Mac OS X v10.2 and later.

Declared in `NSGlyphInfo.h`.

`NSAdobeKorea1CharacterCollection`
Indicates the Adobe-Korea1 mapping.

Available in Mac OS X v10.2 and later.

Declared in `NSGlyphInfo.h`.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSGlyphInfo.h`

NSGradient Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSGradient.h
Companion guide	Cocoa Drawing Guide
Related sample code	CocoaSlides DispatchFractal Grady IconCollection MenuItemView

Overview

The `NSGradient` class provides support for drawing gradient fill colors, also known as shadings in Quartz. This class provides convenience methods for drawing radial or linear (axial) gradients for rectangles and `NSBezierPath` objects. It also supports primitive methods that let you customize the shape of the gradient fill.

A gradient consists of two or more color changes over the range of the gradient shape. When creating a gradient object, you specify the colors and their locations relative to the start and end of the gradient. This combination of color and location is known as a **color stop**. During drawing, the `NSGradient` object uses the color stop information to compute color changes for you and passes that information to the Quartz shading functions.

Because the `NSGradient` class uses Quartz shadings, drawing is handled by computing the colors at a given point mathematically. This technique results in smooth gradients regardless of the resolution of the target device.

For more information about gradients and their appearance, see Gradients in *Quartz 2D Programming Guide*.

Tasks

Initialization

- `initWithStartingColor:endingColor:` (page 1292)
  Initializes a newly allocated gradient object with two colors.
- `initWithColors:` (page 1290)
  Initializes a newly allocated gradient object with an array of colors.
- `initWithColorsAndLocations:` (page 1291)
  Initializes a newly allocated gradient object with a comma-separated list of arguments.
- `initWithColors:atLocations:colorSpace:` (page 1291)
  Initializes a newly allocated gradient object with the specified colors, color locations, and color space.

Primitive Drawing Methods

- `drawFromPoint:toPoint:options:` (page 1286)
  Draws a linear gradient between the specified start and end points.
- `drawFromCenter:radius:toCenter:radius:options:` (page 1285)
  Draws a radial gradient between the specified circles.

Drawing Linear Gradients

- `drawInRect:angle:` (page 1288)
  Fills the specified rectangle with a linear gradient.
- `drawInBezierPath:angle:` (page 1286)
  Fills the specified path with a linear gradient.

Drawing Radial Gradients

- `drawInRect:relativeCenterPosition:` (page 1289)
  Draws a radial gradient starting at the center of the specified rectangle.
- `drawInBezierPath:relativeCenterPosition:` (page 1287)
  Draws a radial gradient starting at the center point of the specified path.

Getting Gradient Properties

- `colorSpace` (page 1285)
  Returns the color space of the colors associated with the receiver.
- `numberOfColorStops` (page 1293)
  Returns the number of color stops associated with the receiver.

- `getColor:location:atIndex:` (page 1289)
  Returns information about the color stop at the specified index in the receiver's color array.
- `interpolatedColorAtLocation:` (page 1292)
  Returns the color of the rendered gradient at the specified relative location.

Instance Methods

colorSpace

Returns the color space of the colors associated with the receiver.

```
- (NSColorSpace *)colorSpace
```

Return Value
The color space object used by the receiver's colors.

Discussion
When the receiver is initialized, colors that do not conform to the receiver's color space are converted automatically.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSGradient.h`

drawFromCenter:radius:toCenter:radius:options:

Draws a radial gradient between the specified circles.

```
- (void)drawFromCenter:(NSPoint)startCenter radius:(CGFloat)startRadius
    toCenter:(NSPoint)endCenter radius:(CGFloat)endRadius
    options:(NSGradientDrawingOptions)options
```

Parameters

startCenter
The center point of the circle that represents the beginning of the gradient.

startRadius
The radius of the circle that represents the beginning of the gradient.

endCenter
The center point of the circle that represents the end of the gradient.

endRadius
The radius of the circle that represents the end of the gradient.

options
The gradient options, if any. You can use these options to extend the gradient size beyond the start and end circles. For more information, see "Gradient Drawing Options" (page 1294).

Discussion
This method draws a radial gradient pattern starting at the first circle and ending at the second circle. The gradient color transitions occur in circular bands emanating from the starting circle and ending at the second circle.

This is a primitive method used by the `NSGradient` class to draw radial gradients. Because this method does not perform any clipping of the gradient fill pattern, you must ensure that the clipping region is configured properly if you intend to invoke this method directly. By default, the clipping region is set to the current view or window in which drawing occurs.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSGradient.h`

drawFromPoint:toPoint:options:

Draws a linear gradient between the specified start and end points.

```
- (void)drawFromPoint:(NSPoint)startingPoint toPoint:(NSPoint)endingPoint
    options:(NSGradientDrawingOptions)options
```

Parameters

startingPoint
The starting point for the gradient, in the local coordinate system. The gradient's first color is drawn at this point.

endingPoint
The end point for the gradient, in the local coordinate system. The gradient's last color is drawn at this point.

options
The gradient options, if any. You can use these options to extend the gradient size beyond the start and end points. For more information, see "Gradient Drawing Options" (page 1294).

Discussion
This method draws the gradient color changes along the line formed by the two points. The gradient fill extends perpendicularly outward from line until it reaches the edges of the current clipping region.

This is a primitive method used by the `NSGradient` class to draw linear gradients. Because this method does not perform any clipping of the gradient fill pattern, you must ensure that the clipping region is configured properly if you intend to invoke this method directly. By default, the clipping region is set to the current view or window in which drawing occurs.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSGradient.h`

drawInBezierPath:angle:

Fills the specified path with a linear gradient.

```
- (void)drawInBezierPath:(NSBezierPath *)path angle:(CGFloat)angle
```

Parameters

path

The path object to fill.

angle

The angle of the linear gradient, specified in degrees. Positive values indicate rotation in the counter-clockwise direction relative to the horizontal axis.

Discussion

This convenience method behaves in a similar way to the `drawInRect:angle:` method, with the path object replacing the rectangle as the clipping region. Like the other method, the start and end colors are guaranteed to be visible at the farthest ends of the path.

The gradient formed by this method is clipped to *path*.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `drawInRect:angle:` (page 1288)
- `drawFromPoint:toPoint:options:` (page 1286)

Related Sample Code

CocoaSlides

Declared In

`NSGradient.h`

drawInBezierPath:relativeCenterPosition:

Draws a radial gradient starting at the center point of the specified path.

```
- (void)drawInBezierPath:(NSBezierPath *)path
    relativeCenterPosition:(NSPoint)relativeCenterPosition
```

Parameters

path

The path to fill.

relativeCenterPosition

The relative location within the bounding rectangle of *path* to use as the center point of the gradient's end circle. Each coordinate must contain a value between -1.0 and 1.0. A coordinate value of 0 represents the center of the path's bounding rectangle along the given axis. In the default coordinate system, a value of -1.0 corresponds to the bottom or left edge of the bounding rectangle and a value of 1.0 corresponds to the top or right edge.

Discussion

The center point of the starting circle is the same as the center point of *path*. The radius of the starting circle is 0, resulting in the starting circle being just a point.

The center point of the end circle starts at the center point of *path* and is modified by the value in the *relativeCenterPosition* parameter. For example, if *relativeCenterPosition* contains the point (1.0, 1.0), the center of the end circle is located in the top-right corner of the path's bounding rectangle. The radius of the end circle is set to the smallest value that ensures `rect` is covered by the end circle.

The gradient formed by this method is clipped to *path*.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `drawInRect:relativeCenterPosition:` (page 1289)
- `drawFromCenter:radius:toCenter:radius:options:` (page 1285)

Declared In
`NSGradient.h`

drawInRect:angle:

Fills the specified rectangle with a linear gradient.

```
- (void)drawInRect:(NSRect)rect angle:(CGFloat)angle
```

Parameters

rect
The rectangle to fill.

angle
The angle of the linear gradient, specified in degrees. Positive values indicate rotation in the counter-clockwise direction relative to the horizontal axis.

Discussion
This convenience method draws a linear gradient inside the specified rectangle. The gradient is drawn so that the start and end colors are guaranteed to be visible in opposite corners of the rectangle. The angle of rotation determines which corner contains the start color; see Table 58-1.

Table 58-1 Linear gradient starting points.

Rotation angle	Start corner
0-89 degrees	Lower-left
90-179 degrees	Lower-right
180-269 degrees	Upper-right
270-359 degrees	Upper-left

The gradient's color transitions occur along the line formed by the angle of rotation. For example, a rotation of 0 degrees results in colors changing from left-to-right across the rectangle, while a rotation of 90 degrees results in colors changing from bottom to top.

The gradient drawn by this method is clipped to *rect*.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `drawInBezierPath:angle:` (page 1286)
- `drawFromPoint:toPoint:options:` (page 1286)

Declared In
`NSGradient.h`

drawInRect:relativeCenterPosition:

Draws a radial gradient starting at the center of the specified rectangle.

```
- (void)drawInRect:(NSRect)rect
    relativeCenterPosition:(NSPoint)relativeCenterPosition
```

Parameters

rect
The rectangle to fill.

relativeCenterPosition
The relative location within the rectangle to use as the center point of the gradient's end circle. Each coordinate must contain a value between -1.0 and 1.0. A coordinate value of 0 represents the center of *rect* along the given axis. In the default coordinate system, a value of -1.0 corresponds to the bottom or left edge of the rectangle and a value of 1.0 corresponds to the top or right edge.

Discussion
The center point of the starting circle is the same as the center point of *rect*. The radius of the starting circle is 0, resulting in the starting circle being just a point.

The center point of the end circle starts at the center point of *rect* and is modified by the value in the *relativeCenterPosition* parameter. For example, if *relativeCenterPosition* contains the point (1.0, 1.0), the center of the end circle is located in the top-right corner of *rect*. The radius of the end circle is set to the smallest value that ensures `rect` is covered by the end circle.

The gradient formed by this method is clipped to *rect*.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `drawInRect:relativeCenterPosition:` (page 1289)
- `drawFromCenter:radius:toCenter:radius:options:` (page 1285)

Declared In
`NSGradient.h`

getColor:location:atIndex:

Returns information about the color stop at the specified index in the receiver's color array.

```
- (void)getColor:(NSColor **)color location:(CGFloat *)location
    atIndex:(NSInteger)index
```

Parameters

color

On input, a pointer to a color object. On output, the color at the specified index in the receiver's color array. You may specify `nil` if you are not interested in this parameter.

location

On input, a pointer to a floating point number. On output, contains the location value associated with the color. This value is between 0.0 and 1.0. It is used to determine the position of the color relative to the start and end points of the gradient. You may specify `NULL` if you are not interested in this parameter.

index

The index of the color you want.

Discussion

This method returns the color stop information that was used to create the receiver. It does not return the interpolated color values at any point along the gradient. The location of the gradient's first color is typically 0.0 and the location of the last color is typically 1.0, although the locations can vary depending on how the receiver was created.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `numberOfColorStops` (page 1293)
- `interpolatedColorAtLocation:` (page 1292)

Declared In

`NSGradient.h`

initWithColors:

Initializes a newly allocated gradient object with an array of colors.

```
- (id)initWithColors:(NSArray *)colorArray
```

Parameters

colorArray

An array of `NSColor` objects representing the colors to use to initialize the gradient. There must be at least two colors in the array. The first color is placed at location 0.0 and the last at location 1.0. If there are more than two colors, the additional colors are placed at evenly spaced intervals between the first and last colors.

Return Value

The initialized `NSGradient` object.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `initWithColors:atLocations:colorSpace:` (page 1291)

Related Sample Code
DispatchFractal

Declared In
NSGradient.h

initWithColors:atLocations:colorSpace:

Initializes a newly allocated gradient object with the specified colors, color locations, and color space.

```
- (id)initWithColors:(NSArray *)colorArray atLocations:(const CGFloat *)locations
    colorSpace:(NSColorSpace *)colorSpace
```

Parameters

colorArray
An array of `NSColor` objects representing the colors in the gradient.

locations
An array of `CGFloat` values containing the location for each color in the gradient. Each value must be in the range 0.0 to 1.0. There must be the same number of locations as are colors in the *colorArray* parameter.

colorSpace
The color space to use for the gradient.

Return Value
The initialized `NSGradient` object.

Discussion
This method is the designated initializer of `NSGradient`. The colors in the *colorArray* parameter are converted to the specified color space if they are not already in that color space.

Typically, at least one color should have a location of 0.0 and one should have a location of 1.0. If these locations are not specified, the color at the closest color stop is used to fill the gap.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSGradient.h

initWithColorsAndLocations:

Initializes a newly allocated gradient object with a comma-separated list of arguments.

```
- (id)initWithColorsAndLocations:(NSColor *)firstColor, ...
```

Parameters

firstColor
The first color in the gradient.

...
A comma-separated list of alternating `NSColor` objects and location arguments (specified as `CGFloat` values). The first value after `firstColor` must be a location. Each location value must be between 0.0 and 1.0. The list must be `nil`-terminated.

Return Value
The initialized `NSGradient` object.

Discussion
Typically, at least one color should have a location of 0.0 and one should have a location of 1.0. If these locations are not specified, the color at the closest color stop is used to fill the gap.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- initWithColors:atLocations:colorSpace:` (page 1291)

Declared In
`NSGradient.h`

initWithStartingColor:endingColor:

Initializes a newly allocated gradient object with two colors.

```
- (id)initWithStartingColor:(NSColor *)startingColor endingColor:(NSColor
    *)endingColor
```

Parameters

startingColor
The starting color of the gradient. The location of this color is fixed at 0.0.

endingColor
The ending color of the gradient. The location of this color is fixed at 1.0.

Return Value
The initialized `NSGradient` object.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- initWithColors:atLocations:colorSpace:` (page 1291)

Related Sample Code
CocoaSlides
Grady
IconCollection
MenuItemView

Declared In
`NSGradient.h`

interpolatedColorAtLocation:

Returns the color of the rendered gradient at the specified relative location.

```
- (NSColor *)interpolatedColorAtLocation:(CGFloat)location
```

Parameters

location

The location value for the color you want. This value must be between 0.0 and 1.0. This value need not correspond to the location of one of the color objects used to create the gradient.

Discussion

This method does not simply return the color values used to initialize the receiver. Instead, it computes the value that would be drawn at the specified location.

The start color of the gradient is always located at 0.0 and the end color is always at 1.0.

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

DispatchFractal

Declared In

`NSGradient.h`

numberOfColorStops

Returns the number of color stops associated with the receiver.

```
- (NSInteger)numberOfColorStops
```

Return Value

The number of colors in the receiver's color array.

Discussion

Gradients must have at least two color stops: one defining the location of the start color and one defining the location of the end color. Gradients may have additional color stops located at different transition points in between the start and end stops.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSGradient.h`

Constants

NSGradientDrawingOptions

Specifies gradient drawing options.

```
typedef NSUInteger NSGradientDrawingOptions;
```

Discussion

The constant values associated with this type are listed in "Gradient Drawing Options" (page 1294).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSGradient.h`

Gradient Drawing Options

These constants are used by the primitive drawing methods to determine if drawing occurs outside of the gradient start and end locations.

```
enum {
    NSGradientDrawsBeforeStartingLocation =   (1 << 0),
    NSGradientDrawsAfterEndingLocation =    (1 << 1),
};
```

Constants

`NSGradientDrawsBeforeStartingLocation`
Drawing extends before the gradient starting point.
Available in Mac OS X v10.5 and later.
Declared in `NSGradient.h`.

`NSGradientDrawsAfterEndingLocation`
Drawing extends beyond the gradient end point.
Available in Mac OS X v10.5 and later.
Declared in `NSGradient.h`.

Declared In
`AppKit/NSGradient.h`

NSGraphicsContext Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSGraphicsContext.h
Companion guide	Cocoa Drawing Guide
Related sample code	FunHouse Quartz EB QuickLookSketch Reducer Sketch-112

Overview

The `NSGraphicsContext` class is the programmatic interface to objects that represent graphics contexts. A context can be thought of as a destination to which drawing and graphics state operations are sent for execution. Each graphics context contains its own graphics environment and state.

The `NSGraphicsContext` class is an abstract superclass for destination-specific graphics contexts. You obtain instances of concrete subclasses with the class methods `currentContext` (page 1298), `graphicsContextWithAttributes:` (page 1299), `graphicsContextWithBitmapImageRep:` (page 1299), `graphicsContextWithGraphicsPort:flipped:` (page 1300), and `graphicsContextWithWindow:` (page 1300).

At any time there is the notion of the current context. The current context for the current thread may be set using `setCurrentContext:` (page 1302).

Graphics contexts are maintained on a stack. You push a graphics context onto the stack by sending it a `saveGraphicsState` (page 1308) message, and pop it off the stack by sending it a `restoreGraphicsState` (page 1307) message. By sending `restoreGraphicsState` (page 1307) to an `NSGraphicsContext` object you remove it from the stack, and the next graphics context on the stack becomes the current graphics context.

Tasks

Creating a Graphics Context

`+ graphicsContextWithAttributes:` (page 1299)
Instantiates and returns an instance of `NSGraphicsContext` using the specified attributes.

`+ graphicsContextWithBitmapImageRep:` (page 1299)
Instantiates and returns a new graphics context using the supplied `NSBitmapImageRep` object as the context destination.

`+ graphicsContextWithGraphicsPort:flipped:` (page 1300)
Instantiates and returns a new graphics context from the given graphics port.

`+ graphicsContextWithWindow:` (page 1300)
Creates and returns a new graphics context for drawing into a window.

Managing the Current Context

`+ currentContext` (page 1298)
Returns the current graphics context of the current thread.

`+ setCurrentContext:` (page 1302)
Sets the current graphics context of the current thread.

`- graphicsPort` (page 1305)
Returns the low-level, platform-specific graphics context represented by the receiver.

Managing the Graphics State

`+ setGraphicsState:` (page 1302)
Makes the graphics context of the specified graphics state current, and resets graphics state.

`+ restoreGraphicsState` (page 1301)
Pops a graphics context from the per-thread stack, makes it current, and sends the context a `restoreGraphicsState` (page 1307) message.

`- restoreGraphicsState` (page 1307)
Removes the receiver's graphics state from the top of the graphics state stack and makes the next graphics state the current graphics state.

`+ saveGraphicsState` (page 1301)
Saves the graphics state of the current graphics context.

`- saveGraphicsState` (page 1308)
Saves the current graphics state and creates a new graphics state on the top of the stack.

Testing the Drawing Destination

`+ currentContextDrawingToScreen` (page 1298)
Returns a Boolean value that indicates whether the current context is drawing to the screen.

- `isDrawingToScreen` (page 1306)
  Returns a Boolean value that indicates whether the drawing destination is the screen.

Getting Information About a Context

- `attributes` (page 1303)
  Returns the receiver's attributes.
- `isFlipped` (page 1306)
  Returns a Boolean value that indicates the receiver's flipped state.

Flushing Graphics to the Context

- `flushGraphics` (page 1305)
  Forces any buffered operations or data to be sent to the receiver's destination.

Managing the Focus Stack

- `focusStack` (page 1305) Available in Mac OS X v10.0 through Mac OS X v10.5
  Returns the object used by the context to track the hierarchy of views with locked focus.
- `setFocusStack:` (page 1309) Available in Mac OS X v10.0 through Mac OS X v10.5
  Sets the object used by the receiver to track the hierarchy of views with locked focus.

Configuring Rendering Options

- `setCompositingOperation:` (page 1309)
  Sets the receiver's global compositing operation.
- `compositingOperation` (page 1304)
  Returns the receiver's global compositing operation setting.
- `setImageInterpolation:` (page 1310)
  Sets the receiver's interpolation behavior.
- `imageInterpolation` (page 1306)
  Returns a constant that specifies the receiver's interpolation behavior.
- `setShouldAntialias:` (page 1311)
  Sets whether the receiver should use antialiasing.
- `shouldAntialias` (page 1311)
  Returns a Boolean value that indicates whether the receiver uses antialiasing.
- `setPatternPhase:` (page 1310)
  Sets the amount to offset the pattern color when filling the receiver.
- `patternPhase` (page 1307)
  Returns the amount to offset the pattern color when filling the receiver.

Getting the Core Image Context

- `CIContext` (page 1303)
  Returns a `CIContext` object that you can use to render into the receiver.

Managing the Color Rendering Intent

- `colorRenderingIntent` (page 1304)
  Returns the current rendering intent in the receiver's graphics state.
- `setColorRenderingIntent:` (page 1308)
  Sets the rendering intent in the receiver's graphics state.

Class Methods

currentContext

Returns the current graphics context of the current thread.

```
+ (NSGraphicsContext *)currentContext
```

Return Value
The current graphics context of the current thread.

Discussion
Returns an instance of a concrete subclass of `NSGraphicsContext`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
FunHouse
Quartz EB
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSGraphicsContext.h`

currentContextDrawingToScreen

Returns a Boolean value that indicates whether the current context is drawing to the screen.

```
+ (BOOL)currentContextDrawingToScreen
```

Return Value
`YES` if the current context is drawing to the screen, otherwise `NO`.

Discussion
This convenience method is equivalent to sending `isDrawingToScreen` (page 1306) to the result of `currentContext` (page 1298).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphicsContext.h`

graphicsContextWithAttributes:

Instantiates and returns an instance of `NSGraphicsContext` using the specified attributes.

```
+ (NSGraphicsContext *)graphicsContextWithAttributes:(NSDictionary *)attributes
```

Parameters

attributes
A dictionary of values associated with the keys described in "Attribute dictionary keys" (page 1312). The attributes specify such things as representation format and destination.

Return Value
A new `NSGraphicsContext` object or `nil` if the object could not be created.

Discussion
Use this method to create a graphics context for a window or bitmap destination. If you want to create a graphics context for a PDF or PostScript destination, do not use this method; instead, use the `NSPrintOperation` class to set up the printing environment needed to generate that type of information.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphicsContext.h`

graphicsContextWithBitmapImageRep:

Instantiates and returns a new graphics context using the supplied `NSBitmapImageRep` object as the context destination.

```
+ (NSGraphicsContext *)graphicsContextWithBitmapImageRep:(NSBitmapImageRep
    *)bitmapRep
```

Parameters

bitmapRep
The `NSBitmapImageRep` object to use as the destination.

Return Value
The created `NSGraphicsContext` object or `nil` if the object could not be created.

Discussion
This method accepts only single plane `NSBitmapImageRep` instances. It is the equivalent of using `graphicsContextWithAttributes:` (page 1299) and passing *bitmapRep* as the value for the dictionary's `NSGraphicsContextDestinationAttributeName` key.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ graphicsContextWithAttributes:` (page 1299)

Related Sample Code
AnimatedTableView
Reducer

Declared In
`NSGraphicsContext.h`

graphicsContextWithGraphicsPort:flipped:

Instantiates and returns a new graphics context from the given graphics port.

```
+ (NSGraphicsContext *)graphicsContextWithGraphicsPort:(void *)graphicsPort
    flipped:(BOOL)initialFlippedState
```

Parameters

graphicsPort
The graphics port used to create the graphics-context object. Typically `graphicsPort` is a `CGContextRef` (opaque type) object.

initialFlippedState
Specifies the receiver's initial flipped state. This is the value returned by `isFlipped` (page 1306) when no view has focus.

Return Value
The created `NSGraphicsContext` object or `nil` if the object could not be created.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
CIAnnotation
CIVideoDemoGL
FunHouse
iChatTheater
QuickLookSketch

Declared In
`NSGraphicsContext.h`

graphicsContextWithWindow:

Creates and returns a new graphics context for drawing into a window.

```
+ (NSGraphicsContext *)graphicsContextWithWindow:(NSWindow *)aWindow
```

Parameters

aWindow

The window object representing the window to use for drawing.

Return Value

The created `NSGraphicsContext` object or `nil` if the object could not be created.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

JAWTExample

Declared In

`NSGraphicsContext.h`

restoreGraphicsState

Pops a graphics context from the per-thread stack, makes it current, and sends the context a `restoreGraphicsState` (page 1307) message.

```
+ (void)restoreGraphicsState
```

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

QuickLookSketch

Reducer

Sketch+Accessibility

Sketch-112

TrackBall

Declared In

`NSGraphicsContext.h`

saveGraphicsState

Saves the graphics state of the current graphics context.

```
+ (void)saveGraphicsState
```

Discussion

This method sends the current graphics context a `saveGraphicsState` (page 1308) message and pushes the context onto the per-thread stack.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

QuickLookSketch

Reducer

Sketch+Accessibility
Sketch-112
TrackBall

Declared In
`NSGraphicsContext.h`

setCurrentContext:

Sets the current graphics context of the current thread.

```
+ (void)setCurrentContext:(NSGraphicsContext *)context
```

Parameters

context

The graphics-context object to set as the current one. This must be an instance of a concrete subclass of `NSGraphicsContext`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CIAnnotation
CIVideoDemoGL
FunHouse
iChatTheater
QuickLookSketch

Declared In
`NSGraphicsContext.h`

setGraphicsState:

Makes the graphics context of the specified graphics state current, and resets graphics state.

```
+ (void)setGraphicsState:(NSInteger)graphicsState
```

Discussion
The *graphicState* identifier must be created in the calling thread.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphicsContext.h`

Instance Methods

attributes

Returns the receiver's attributes.

```
- (NSDictionary *)attributes
```

Return Value
The receiver's attributes, if any.

Discussion
Screen-based graphics contexts do not store attributes, even if you create them using `graphicsContextWithAttributes:` (page 1299).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphicsContext.h`

CIContext

Returns a `CIContext` object that you can use to render into the receiver.

```
- (CIContext *)CIContext
```

Return Value
A `CIContext` object or `nil` if the object could not be created.

Discussion
The `CIContext` object is created on demand and remains in existence for the lifetime of its owning `NSGraphicsContext` object. A `CIContext` object is an evaluation context for rendering a `CIImage` object through Quartz 2D or OpenGL. You use `CIContext`objects in conjunction with `CIFilter`, `CIImage`, `CIVector`, and `CIColor` objects to take advantage of the built-in Core Image filters when processing images.

For more on `CIContext` objects and related Core Image objects, see *Core Image Programming Guide*.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
AnimatedTableView
CIHazeFilterSample
CITransitionSelectorSample
FunHouse
Reducer

Declared In
`NSGraphicsContext.h`

colorRenderingIntent

Returns the current rendering intent in the receiver's graphics state.

```
- (NSColorRenderingIntent)colorRenderingIntent
```

Return Value
An "Creating a Graphics Context" (page 1296)value that specifies the rendering intent currently used by the receiver. For possible values see "Color Rendering Intent Constants" (page 1313).

Discussion
The rendering intent specifies how Cocoa should handle colors that are not located within the gamut of the destination color space of a graphics context.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setColorRenderingIntent: (page 1308)

Declared In
NSGraphicsContext.h

compositingOperation

Returns the receiver's global compositing operation setting.

```
- (NSCompositingOperation)compositingOperation
```

Return Value
The receiver's global compositing operation setting. See NSCompositingOperation (page 1375) for valid constants.

Discussion
The compositing operation is a global attribute of the graphics context and affects drawing operations that do not take an explicit compositing operation parameter. For methods that do take an explicit compositing operation parameter, the value of that parameter supersedes the global value.

The compositing operations are related to (but different from) the blend mode settings used in Quartz. Only the default compositing operation (NSCompositeCopy) is supported for PDF or PostScript content.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setCompositingOperation: (page 1309)

Related Sample Code
ImageMap
ImageMapExample

Declared In
NSGraphicsContext.h

flushGraphics

Forces any buffered operations or data to be sent to the receiver's destination.

```
- (void)flushGraphics
```

Discussion
Graphics contexts use buffers to queue pending operations but for efficiency reasons may not always empty those buffers immediately. This method forces the buffers to be emptied.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaAUHost
iChatTheater

Declared In
`NSGraphicsContext.h`

focusStack

Returns the object used by the context to track the hierarchy of views with locked focus. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (void *)focusStack
```

Return Value
The object used by the context to track the hierarchy of views with locked focus.

Discussion
You should never need to get or modify the focus stack information. The use of focus stacks may be deprecated in a future release.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared In
`NSGraphicsContext.h`

graphicsPort

Returns the low-level, platform-specific graphics context represented by the receiver.

```
- (void *)graphicsPort
```

Discussion
In Mac OS X, this is the Core Graphics context, a `CGContextRef` object (opaque type).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
FunHouse

ImageApp
Quartz EB
UnsharpMask
WebKitDOMElementPlugIn

Declared In
NSGraphicsContext.h

imageInterpolation

Returns a constant that specifies the receiver's interpolation behavior.

```
- (NSImageInterpolation)imageInterpolation
```

Return Value
The receiver's interpolation (image smoothing) behavior.

Discussion
The NSImageInterpolation constants are described in NSImageInterpolation (page 1313).

Availability
Available in Mac OS X v10.0 and later.

See Also
- setImageInterpolation: (page 1310)

Declared In
NSGraphicsContext.h

isDrawingToScreen

Returns a Boolean value that indicates whether the drawing destination is the screen.

```
- (BOOL)isDrawingToScreen
```

Return Value
YES if the drawing destination is the screen, otherwise NO.

Discussion
A return value of NO may mean that the drawing destination is a printer, but the destination may also be a PDF or EPS file. If this method returns NO, you can call attributes (page 1303) to see if additional information is available about the drawing destination.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSGraphicsContext.h

isFlipped

Returns a Boolean value that indicates the receiver's flipped state.

```
- (BOOL)isFlipped
```

Return Value
`YES` if the receiver is flipped, otherwise `NO`.

Discussion
The state is determined by sending `isFlipped` to the receiver's view that has focus. If no view has focus, returns `NO` unless the receiver is instantiated using `graphicsContextWithGraphicsPort:flipped:` (page 1300) specifying `YES` as the flipped parameter.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ graphicsContextWithGraphicsPort:flipped:` (page 1300)

Declared In
`NSGraphicsContext.h`

patternPhase

Returns the amount to offset the pattern color when filling the receiver.

```
- (NSPoint)patternPhase
```

Return Value
The amount to offset the pattern color when filling the receiver.

Discussion
The pattern phase is a translation (width, height) applied before a pattern is drawn in the current context and is part of the saved graphics state of the context. The default pattern phase is (0,0). Setting the pattern phase has the effect of temporarily changing the pattern matrix of any pattern you decide to draw. For example, setting the pattern phase to (2,3) has the effect of moving the start of pattern cell tiling to the point (2,3) in default user space.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- setPatternPhase:` (page 1310)

Declared In
`NSGraphicsContext.h`

restoreGraphicsState

Removes the receiver's graphics state from the top of the graphics state stack and makes the next graphics state the current graphics state.

```
- (void)restoreGraphicsState
```

Discussion
This method must have been preceded with a `saveGraphicsState` (page 1308) message to add the graphics state to the stack. Invocations of `saveGraphicsState` and `restoreGraphicsState` methods may be nested.

Restoring the graphics state restores such attributes as the current drawing style, transformation matrix, color, and font of the original graphics state.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphicsContext.h`

saveGraphicsState

Saves the current graphics state and creates a new graphics state on the top of the stack.

```
- (void)saveGraphicsState
```

Discussion
The new graphics state is a copy of the previous state that can be modified to handle new drawing operations.

Saving the graphics state saves such attributes as the current drawing style, transformation matrix, color, and font. To set drawing style attributes, use the methods of `NSBezierPath`. Other attributes are accessed through appropriate objects such as `NSAffineTransform`, `NSColor`, and `NSFont`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphicsContext.h`

setColorRenderingIntent:

Sets the rendering intent in the receiver's graphics state.

```
- (void)setColorRenderingIntent:(NSColorRenderingIntent)renderingIntent
```

Parameters
renderingIntent
An "Creating a Graphics Context" (page 1296)value that specifies the rendering intent to be used. For possible values see "NSColorRenderingIntent" (page 1313).

Discussion
The rendering intent specifies how Cocoa should handle colors that are not located within the gamut of the destination color space of a graphics context. If you do not explicitly set the rendering intent, and sampled images are being drawn, `NSGraphicsContext` uses perceptual rendering intent. Otherwise, `NSGraphicsContext` uses relative colorimetric rendering intent

Availability
Available in Mac OS X v10.5 and later.

See Also
- `colorRenderingIntent` (page 1304)

Declared In
`NSGraphicsContext.h`

setCompositingOperation:

Sets the receiver's global compositing operation.

```
- (void)setCompositingOperation:(NSCompositingOperation)operation
```

Parameters

operation

A constant that specifies a compositing operating. See `NSCompositingOperation` (page 1375) for valid constants.

Discussion
The compositing operation is a global attribute of the graphics context and affects drawing operations that do not take an explicit compositing operation parameter. For methods that do take an explicit compositing operation parameter, the value of that parameter supersedes the global value.

The compositing operations are related to (but different from) the blend mode settings used in Quartz. Only the default compositing operation (`NSCompositeCopy`) is supported when rendering PDF or PostScript content.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `compositingOperation` (page 1304)

Related Sample Code
ImageMap

ImageMapExample

Declared In
`NSGraphicsContext.h`

setFocusStack:

Sets the object used by the receiver to track the hierarchy of views with locked focus. (Available in Mac OS X v10.0 through Mac OS X v10.5.)

```
- (void)setFocusStack:(void *)stack
```

Parameters

stack

The object used by the graphics context for view-hierarchy tracking.

Discussion
You should never need to get or modify the focus stack information. The use of focus stacks may be deprecated in a future release.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared In
NSGraphicsContext.h

setImageInterpolation:

Sets the receiver's interpolation behavior.

```
- (void)setImageInterpolation:(NSImageInterpolation)interpolation
```

Parameters

interpolation
A constant specifying the image-interpolation behavior. The `NSImageInterpolation` constants are described in `NSImageInterpolation` (page 1313).

Discussion
Note that this value is not part of the graphics state, so it cannot be reset using `restoreGraphicsState` (page 1307).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- imageInterpolation` (page 1306)

Related Sample Code
WebKitDOMElementPlugIn

Declared In
NSGraphicsContext.h

setPatternPhase:

Sets the amount to offset the pattern color when filling the receiver.

```
- (void)setPatternPhase:(NSPoint)phase
```

Parameters

phase
A point specifying the offset.

Discussion
Use this method when you need to line up the pattern color with another pattern, such as the pattern in a superview.

The pattern phase is a translation (width, height) applied before a pattern is drawn in the current context and is part of the saved graphics state of the context. The default pattern phase is (0,0). Setting the pattern phase has the effect of temporarily changing the pattern matrix of any pattern you decide to draw. For example, setting the pattern phase to (2,3) has the effect of moving the start of pattern cell tiling to the point (2,3) in default user space.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `patternPhase` (page 1307)

Declared In
`NSGraphicsContext.h`

setShouldAntialias:

Sets whether the receiver should use antialiasing.

```
- (void)setShouldAntialias:(BOOL)antialias
```

Parameters

antialias
`YES` if the receiver should use antialiasing, otherwise `NO`.

Discussion
This value is part of the graphics state and is restored by `restoreGraphicsState` (page 1307).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `shouldAntialias` (page 1311)

Related Sample Code
Cocoa OpenGL
Cropped Image
From A View to A Movie
From A View to A Picture

Declared In
`NSGraphicsContext.h`

shouldAntialias

Returns a Boolean value that indicates whether the receiver uses antialiasing.

```
- (BOOL)shouldAntialias
```

Return Value
`YES` if the receiver uses antialiasing, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setShouldAntialias:` (page 1311)

Declared In
NSGraphicsContext.h

Constants

Attribute dictionary keys

These constants are dictionary keys used by graphicsContextWithAttributes: (page 1299) and attributes (page 1303).

```
NSString *NSGraphicsContextDestinationAttributeName;
NSString *NSGraphicsContextRepresentationFormatAttributeName;
```

Constants

NSGraphicsContextDestinationAttributeName
Can be an instance of NSWindow or NSBitmapImageRep when creating a graphics context.

When determining the type of a graphics context, this value can be an NSMutableData, NSString, or NSURL object.

Available in Mac OS X v10.0 and later.

Declared in NSGraphicsContext.h.

NSGraphicsContextRepresentationFormatAttributeName
Specifies the destination file format.

This value should be retrieved only and not used to create a graphics context.

Available in Mac OS X v10.0 and later.

Declared in NSGraphicsContext.h.

Representation format attribute keys

These constants are possible values for the NSGraphicsContextRepresentationFormatAttributeName key in a graphic context's attribute dictionary.

```
NSString *NSGraphicsContextPSFormat;
NSString *NSGraphicsContextPDFFormat;
```

Constants

NSGraphicsContextPDFFormat
Destination file format is PDF.

Available in Mac OS X v10.0 and later.

Declared in NSGraphicsContext.h.

NSGraphicsContextPSFormat
Destination file format is PostScript.

Available in Mac OS X v10.0 and later.

Declared in NSGraphicsContext.h.

NSImageInterpolation

These interpolations are used by `imageInterpolation` (page 1306) and `setImageInterpolation:` (page 1310).

```
enum {
    NSImageInterpolationDefault,
    NSImageInterpolationNone,
    NSImageInterpolationLow,
    NSImageInterpolationMedium,
    NSImageInterpolationHigh
};
typedef NSUInteger NSImageInterpolation;
```

Constants

`NSImageInterpolationDefault`
Use the context's default interpolation.
Available in Mac OS X v10.0 and later.
Declared in `NSGraphicsContext.h`.

`NSImageInterpolationNone`
No interpolation.
Available in Mac OS X v10.0 and later.
Declared in `NSGraphicsContext.h`.

`NSImageInterpolationLow`
Fast, low-quality interpolation.
Available in Mac OS X v10.0 and later.
Declared in `NSGraphicsContext.h`.

`NSImageInterpolationMedium`
Medium quality, slower than `NSImageInterpolationLow`.
Available in Mac OS X v10.6 and later.
Declared in `NSGraphicsContext.h`.

`NSImageInterpolationHigh`
Slower, higher-quality interpolation.
Available in Mac OS X v10.0 and later.
Declared in `NSGraphicsContext.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphicsContext.h`

NSColorRenderingIntent

These constants specify how Cocoa should handle colors that are not located within the destination color space of a graphics context. These constants are used by the methods `setColorRenderingIntent:` (page 1308) and `colorRenderingIntent` (page 1304).

```
enum {
    NSColorRenderingIntentDefault,
    NSColorRenderingIntentAbsoluteColorimetric,
    NSColorRenderingIntentRelativeColorimetric,
    NSColorRenderingIntentPerceptual,
    NSColorRenderingIntentSaturation
};
typedef NSInteger NSColorRenderingIntent;
```

Constants

`NSColorRenderingIntentDefault`

Use the default rendering intent for the graphics context.

Available in Mac OS X v10.5 and later.

Declared in `NSGraphics.h`.

`NSColorRenderingIntentAbsoluteColorimetric`

Map colors outside of the gamut of the output device to the closest possible match inside the gamut of the output device.

This operation can produce a clipping effect, where two different color values in the gamut of the graphics context are mapped to the same color value in the output device's gamut. Unlike the relative colorimetric, absolute colorimetric does not modify colors inside the gamut of the output device.

Available in Mac OS X v10.5 and later.

Declared in `NSGraphics.h`.

`NSColorRenderingIntentRelativeColorimetric`

Map colors outside of the gamut of the output device to the closest possible match inside the gamut of the output device.

This operation can produce a clipping effect, where two different color values in the gamut of the graphics context are mapped to the same color value in the output device's gamut. The relative colorimetric shifts all colors (including those within the gamut) to account for the difference between the white point of the graphics context and the white point of the output device.

Available in Mac OS X v10.5 and later.

Declared in `NSGraphics.h`.

`NSColorRenderingIntentPerceptual`

Preserve the visual relationship between colors by compressing the gamut of the graphics context to fit inside the gamut of the output device.

Perceptual intent is good for photographs and other complex, detailed images.

Available in Mac OS X v10.5 and later.

Declared in `NSGraphics.h`.

`NSColorRenderingIntentSaturation`

Preserve the relative saturation value of the colors when converting into the gamut of the output device.

The result is an image with bright, saturated colors. Saturation intent is good for reproducing images with low detail, such as presentation charts and graphs.

Available in Mac OS X v10.5 and later.

Declared in `NSGraphics.h`.

NSHelpManager Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSHelpManager.h
Companion guide	Online Help

Overview

The `NSHelpManager` class provides an approach to displaying online help. An application contains one `NSHelpManager` object.

Tasks

Getting the Help Manager

`+ sharedHelpManager` (page 1317)
Returns the shared `NSHelpManager` (page 1315) instance, creating it if it does not already exist.

Displaying Help

`- findString:inBook:` (page 1318)
Performs a search for the specified string in the specified book.

`- openHelpAnchor:inBook:` (page 1318)
Finds and displays the text at the given anchor location in the given book.

Dynamically Adding Help Books

`- registerBooksInBundle:` (page 1319)
Registers one or more help books in the given bundle.

Configuring Context-Sensitive Help

`+ isContextHelpModeActive` (page 1316)
Indicates whether context-sensitive help mode is active.

`+ setContextHelpModeActive:` (page 1316)
Specifies whether context-sensitive help mode is active.

`- setContextHelp:forObject:` (page 1320)
Associates help content with an object.

`- removeContextHelpForObject:` (page 1319)
Removes the association between an object and its context-sensitive help.

Displaying Context-Sensitive Help

`- contextHelpForObject:` (page 1317)
Returns context-sensitive help for an object.

`- showContextHelpForObject:locationHint:` (page 1320)
Displays the context-sensitive help for a given object at or near the point on the screen specified by a given point.

Class Methods

isContextHelpModeActive

Indicates whether context-sensitive help mode is active.

```
+ (BOOL)isContextHelpModeActive
```

Return Value
`YES` when the application is in context-sensitive help mode, `NO` otherwise.

Discussion
In context-sensitive help mode, when a user clicks a user interface item, help for that item is displayed in a small window just below the cursor.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ setContextHelpModeActive:` (page 1316)

Declared In
`NSHelpManager.h`

setContextHelpModeActive:

Specifies whether context-sensitive help mode is active.

```
+ (void)setContextHelpModeActive:(BOOL)contextHelpActive
```

Parameters

contextHelpActive

YES turns on context-sensitive help, NO turns it off.

Discussion

You never send this message directly; instead, the NSApplication method activateContextHelpMode: (page 136) activates context-sensitive help mode, and the first mouse click after displaying the context-sensitive help window deactivates it.

When the application enters context-sensitive help mode, the help manager posts an NSContextHelpModeDidActivateNotification (page 1321) to the default notification center. When the application returns to normal operation, the help manager posts an NSContextHelpModeDidDeactivateNotification (page 1321).

Availability

Available in Mac OS X v10.0 and later.

See Also

+ isContextHelpModeActive (page 1316)

Declared In

NSHelpManager.h

sharedHelpManager

Returns the shared NSHelpManager (page 1315) instance, creating it if it does not already exist.

```
+ (NSHelpManager *)sharedHelpManager
```

Return Value

Shared help manager.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSHelpManager.h

Instance Methods

contextHelpForObject:

Returns context-sensitive help for an object.

```
- (NSAttributedString *)contextHelpForObject:(id)object
```

Parameters

object

Object for which context-sensitive help is sought.

Return Value
Context-sensitive help content.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setContextHelp:forObject:` (page 1320)
- `showContextHelpForObject:locationHint:` (page 1320)

Declared In
`NSHelpManager.h`

findString:inBook:

Performs a search for the specified string in the specified book.

```
- (void)findString:(NSString *)query inBook:(NSString *)book
```

Parameters

query
String to search for.

book
Localized help book to search. When `nil`, all installed help books are searched.

Discussion
To search for a string in your bundle's localized help book, you could use code similar to the following:

```
NSString *locBookName = [[NSBundle mainBundle] objectForInfoDictionaryKey:
@"CFBundleHelpBookName"];
[[NSHelpManager sharedHelpManager] findString:@"Hello"  inBook:locBookName];
```

This is a wrapper for `AHRegisterHelpBook` (which is called only once to register the help book specified in the application's main bundle) and `AHSearch`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSHelpManager.h`

openHelpAnchor:inBook:

Finds and displays the text at the given anchor location in the given book.

```
- (void)openHelpAnchor:(NSString *)anchor inBook:(NSString *)book
```

Parameters

anchor
Location of the desired text.

book
Help book containing the anchor. When `nil`, all installed help books are searched.

Discussion
To open an anchor in your bundle's localized help book, you could use code similar to the following:

```
NSString *locBookName = [[NSBundle mainBundle] objectForInfoDictionaryKey:
@"CFBundleHelpBookName"];
[[NSHelpManager sharedHelpManager] openHelpAnchor:@"anchor1"  inBook:locBookName];
```

This method is a wrapper for `AHRegisterHelpBook` (which is called only once to register the help book specified in the application's main bundle) and `AHLookupAnchor`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSHelpManager.h`

registerBooksInBundle:

Registers one or more help books in the given bundle.

```
- (BOOL)registerBooksInBundle:(NSBundle *)bundle
```

Parameters
bundle
The bundle for additional help books. Books in the main bundle are automatically registered.

Return Value
`YES` if registration is successful, `NO` if if the bundle doesn't contain any help books or if registration fails.

Discussion
You use `registerBooksInBundle:` to register help books in, for example, a plug-in bundle. The `Info.plist` in the bundle should contain a help book directory path, which specifies one or more folders containing help books.

The main bundle is automatically registered by `openHelpAnchor:inBook:` (page 1318) and `findString:inBook:` (page 1318).

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSHelpManager.h`

removeContextHelpForObject:

Removes the association between an object and its context-sensitive help.

```
- (void)removeContextHelpForObject:(id)object
```

Parameters
object
Object to disassociate from its help content.

Discussion
If *object* does not have context-sensitive help associated with it, this method does nothing.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setContextHelp:forObject: (page 1320)

Declared In
NSHelpManager.h

setContextHelp:forObject:

Associates help content with an object.

```
- (void)setContextHelp:(NSAttributedString *)help forObject:(id)object
```

Parameters

help
Help content to associate with *object*.

object
Object to associate with *help*.

Discussion
When the application enters context-sensitive help mode, if *object* is clicked, *help* appears in the context-sensitive help window.

Availability
Available in Mac OS X v10.0 and later.

See Also
- removeContextHelpForObject: (page 1319)

Declared In
NSHelpManager.h

showContextHelpForObject:locationHint:

Displays the context-sensitive help for a given object at or near the point on the screen specified by a given point.

```
- (BOOL)showContextHelpForObject:(id)object locationHint:(NSPoint)point
```

Parameters

object
Object for which context-sensitive help is sought.

point
Screen location at which to display the help content; it's usually under the cursor.

Return Value
YES when help content is successfully displayed. NO if help content is not displayed (for example, when there is no context-sensitive help associated with *object*).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- contextHelpForObject:` (page 1317)

Declared In
`NSHelpManager.h`

Notifications

NSContextHelpModeDidActivateNotification

Posted when the application enters context-sensitive help mode. This typically happens when the user holds down the Help key.

The notification object is the help manager. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSHelpManager.h`

NSContextHelpModeDidDeactivateNotification

Posted when the application exits context-sensitive help mode. This happens when the user clicks the mouse button while the cursor is anywhere on the screen after displaying a context-sensitive help topic.

The notification object is the help manager. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSHelpManager.h`

NSImage Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSPasteboardWriting NSPasteboardReading NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSImage.h
Companion guide	Cocoa Drawing Guide
Related sample code	AnimatedTableView ImageBackground MyPhoto RGB Image Sketch-112

Overview

An `NSImage` object is a high-level class for manipulating image data. You use this class to load existing images or create new ones and composite them into a view or other image. This class works in conjunction with one or more image representation objects (subclasses of `NSImageRep`), which manage the actual image data.

Tasks

Initializing a New NSImage Object

`- initByReferencingFile:` (page 1350)
Initializes and returns an `NSImage` instance and associates it with the specified file.

`- initByReferencingURL:` (page 1351)
Initializes and returns an `NSImage` instance and associates it with the specified URL.

`- initWithCGImage:size:` (page 1351)
Initializes and returns an `NSImage` instance with the contents of the CGImage.

- `initWithContentsOfFile:` (page 1352)
  Initializes and returns an `NSImage` instance with the contents of the specified file.
- `initWithContentsOfURL:` (page 1352)
  Initializes and returns an `NSImage` instance with the contents of the specified URL.
- `initWithData:` (page 1353)
  Initializes and returns an `NSImage` instance with the contents of the specified `NSData` object.
- `initWithDataIgnoringOrientation:` (page 1353)
  Initializes and returns an `NSImage` instance with the contents of the specified `NSData` object, ignoring the EXIF orientation tags..
- `initWithPasteboard:` (page 1354)
  Initializes and returns an `NSImage` instance with data from the specified pasteboard.
- `initWithSize:` (page 1355)
  Initializes and returns an `NSImage` instance whose size is set to the specified value.
- `initWithIconRef:` (page 1354)
  Initializes the image object with a Carbon-style icon resource.

Setting the Image Attributes

- `setSize:` (page 1370)
  Sets the width and height of the image.
- `size` (page 1372)
  Returns the size of the receiver.
- `isTemplate` (page 1357)
  Returns a Boolean value indicating whether the image is a template image.
- `setTemplate:` (page 1371)
  Sets whether the image represents a template image.

Referring to Images by Name

+ `imageNamed:` (page 1330)
  Returns the `NSImage` instance associated with the specified name.
- `setName:` (page 1368)
  Registers the receiver under the specified name.
- `name` (page 1360)
  Returns the name associated with the receiver, if any.

Determining the Supported Image Types

+ `canInitWithPasteboard:` (page 1329)
  Tests whether the receiver can create an instance of itself using pasteboard data.
+ `imageTypes` (page 1332)
  Returns an array of UTI strings identifying the image types supported by the registered `NSImageRep` objects, either directly or through a user-installed filter service.

+ `imageUnfilteredTypes` (page 1334)
  Returns an array of UTI strings identifying the image types supported directly by the registered `NSImageRep` objects.
+ `imageFileTypes` (page 1330)
  Returns an array of strings identifying the image types supported by the registered `NSImageRep` objects.
+ `imageUnfilteredFileTypes` (page 1333)
  Returns an array of strings identifying the file types supported directly by the registered `NSImageRep` objects.
+ `imagePasteboardTypes` (page 1332)
  Returns an array of strings identifying the pasteboard types supported directly by the registered `NSImageRep` objects.
+ `imageUnfilteredPasteboardTypes` (page 1333)
  Returns an array of strings identifying the pasteboard types supported directly by the registered `NSImageRep` objects.

Working With Image Representations

- `addRepresentation:` (page 1335)
  Adds the specified image representation object to to the receiver.
- `addRepresentations:` (page 1335)
  Adds an array of image representation objects to the receiver.
- `representations` (page 1362)
  Returns an array containing all of the receiver's image representations.
- `removeRepresentation:` (page 1362)
  Removes the specified image representation from the receiver and releases it.
- `bestRepresentationForRect:context:hints:` (page 1338)
  Returns the best representation of the image for the specified rect using the provided hints.
- `bestRepresentationForDevice:` (page 1337) Deprecated in Mac OS X v10.6
  Returns the best representation for the device with the specified characteristics.

Hit Testing an Image

- `hitTestRect:withImageDestinationRect:context:hints:flipped:` (page 1349)
  Returns whether the destination rectangle would intersect a non-transparent portion of the image.

Setting the Image Representation Selection Criteria

- `setPrefersColorMatch:` (page 1369)
  Sets whether choosing an image representation favors color matching over resolution matching.
- `prefersColorMatch` (page 1361)
  Returns a Boolean value indicating whether the image prefers to choose image representations using color matching or resolution matching.

- `setUsesEPSOnResolutionMismatch:` (page 1371)
  Sets whether EPS image representations are preferred when no other representations match the resolution of the device.
- `usesEPSOnResolutionMismatch` (page 1374)
  Returns a Boolean value indicating whether EPS representations are preferred when no other representations match the resolution of the device.
- `setMatchesOnMultipleResolution:` (page 1368)
  Sets whether image representations whose resolutions are integral multiples of the device resolution are considered a match.
- `matchesOnMultipleResolution` (page 1360)
  Returns a Boolean value indicating whether image representations whose resolution is an integral multiple of the device resolution are considered a match.

Managing the Focus

- `lockFocus` (page 1358)
  Prepares the image to receive drawing commands.
- `lockFocusFlipped:` (page 1359)
  Prepares the image to receive drawing commands using the specified flipped state.
- `unlockFocus` (page 1374)
  Removes the focus from the receiver.
- `lockFocusOnRepresentation:` (page 1359) Deprecated in Mac OS X v10.6
  Prepares the specified image representation to receive drawing commands. (Deprecated. Use the code fragment shown in the special considerations below.)

Drawing the Image

- `drawAtPoint:fromRect:operation:fraction:` (page 1346)
  Draws all or part of the image at the specified point in the current coordinate system.
- `drawInRect:fromRect:operation:fraction:` (page 1347)
  Draws all or part of the image in the specified rectangle in the current coordinate system.
- `drawRepresentation:inRect:` (page 1348)
  Draws the image using the specified image representation object.
- `drawInRect:fromRect:operation:fraction:respectFlipped:hints:` (page 1347)
  Draws all or part of the image in the specified rectangle respecting the flippedness and hints.
- `compositeToPoint:fromRect:operation:` (page 1340) Deprecated in Mac OS X v10.6
  Composites a portion of the image to the specified point in the current coordinate system.
- `compositeToPoint:fromRect:operation:fraction:` (page 1341) Deprecated in Mac OS X v10.6
  Composites a portion of the image at the specified opacity to the current coordinate system.
- `compositeToPoint:operation:` (page 1342) Deprecated in Mac OS X v10.6
  Composites the entire image to the specified point in the current coordinate system.
- `compositeToPoint:operation:fraction:` (page 1343) Deprecated in Mac OS X v10.6
  Composites the entire image at the specified opacity in the current coordinate system.

- `dissolveToPoint:fraction:` (page 1344) Deprecated in Mac OS X v10.6
  Composites the entire image to the specified location using the `NSCompositeSourceOver` operator.
- `dissolveToPoint:fromRect:fraction:` (page 1345) Deprecated in Mac OS X v10.6
  Composites a portion of the image to the specified location using the `NSCompositeSourceOver` operator.

Working With Alignment Metadata

- `alignmentRect` (page 1336)
  Returns alignment metadata that your code can use to position the image during layout.
- `setAlignmentRect:` (page 1363)
  Sets the alignment metadata that your code can use to position the image during layout.

Setting the Image Storage Options

- `cacheMode` (page 1339)
  Returns the receiver's caching mode.
- `setCacheMode:` (page 1365)
  Set the receiver's caching mode.
- `cacheDepthMatchesImageDepth` (page 1338) Deprecated in Mac OS X v10.6
  Returns a Boolean value indicating whether an image's offscreen window caches use the same bit depth as the image data itself. (Deprecated. `NSImage` no longer caches to windows. A cache is now generated appropriate for the destination where an image is drawn. There is no replacement method.)
- `isCachedSeparately` (page 1356) Deprecated in Mac OS X v10.6
  Returns a Boolean value indicating whether each image representation caches its contents in a separate offscreen window. (Deprecated. NSImage no longer caches to windows. There is no replacement method)
- `isDataRetained` (page 1356) Deprecated in Mac OS X v10.6
  Returns a Boolean value indicating whether the receiver retains its source image data. (Deprecated. In Mac OS v10.6, `NSImage` no longer discards data in such a way that the original can no longer be reconstructed. There is no replacement method.)
- `setCacheDepthMatchesImageDepth:` (page 1364) Deprecated in Mac OS X v10.6
  Sets whether the receiver's offscreen window caches use the same bit depth as the image data itself. (Deprecated. `NSImage` no longer caches to windows. A cache is now generated appropriate for the destination where an image is drawn. There is no replacement method.)
- `setCachedSeparately:` (page 1365) Deprecated in Mac OS X v10.6
  Sets whether each image representation uses a separate offscreen window to cache its contents. (Deprecated. `NSImage` no longer caches to windows. There is no replacement method)
- `setDataRetained:` (page 1366) Deprecated in Mac OS X v10.6
  Sets whether the receiver retains its source image data. (Deprecated. In Mac OS v10.6, `NSImage` no longer discards data in such a way that the original can no longer be reconstructed. There is no replacement method.)

Setting the Image Drawing Options

- `isValid` (page 1357)
  Returns a Boolean value indicating whether an image representation from the receiver can be drawn.
- `setBackgroundColor:` (page 1364)
  Sets the background color of the image.
- `backgroundColor` (page 1336)
  Returns the background color of image.
- `isFlipped` (page 1357)
  Returns a Boolean value indicating whether the image uses a flipped coordinate system. (Deprecated. The flipped property of an image was widely misunderstood and has been deprecated. Use `drawInRect:fromRect:operation:fraction:respectFlipped:hints:` (page 1347) to draw respecting a context's flipped status and `lockFocusFlipped:` (page 1359) to draw into a flipped image.)
- `recache` (page 1361)
  Invalidates and frees the offscreen caches of all image representations.
- `scalesWhenResized` (page 1362) Deprecated in Mac OS X v10.6
  Returns a Boolean value indicating whether image representations are scaled to fit the receiver's size. (Deprecated. This method was related to caching behavior. In Mac OS X v10.6 and later image caching is no long necessary and as a resuilt there is no replacement necessary.)
- `setFlipped:` (page 1367) Deprecated in Mac OS X v10.6
  Sets whether the polarity of the y axis is inverted when drawing an image. (Deprecated. The flipped property of an image was widely misunderstood and has been deprecated. Use `drawInRect:fromRect:operation:fraction:respectFlipped:hints:` (page 1347) to draw respecting a context's flipped status and `lockFocusFlipped:` (page 1359) to draw into a flipped image.)
- `setScalesWhenResized:` (page 1369) Deprecated in Mac OS X v10.6
  Sets whether different-sized image representations are scaled to fit the receiver's size. (Deprecated. This method was related to caching behavior. In Mac OS X v10.6 and later image caching is no long necessary and as a resuilt there is no replacement necessary.)

Assigning a Delegate

- `setDelegate:` (page 1366)
  Sets the delegate object of the receiver.
- `delegate` (page 1344)
  Returns the delegate object of the receiver

Producing TIFF Data for the Image

- `TIFFRepresentation` (page 1372)
  Returns a data object containing TIFF data for all of the image representations in the receiver.
- `TIFFRepresentationUsingCompression:factor:` (page 1373)
  Returns a data object containing TIFF data with the specified compression settings for all of the image representations in the receiver.

Producing a CGImage from an Image

- `CGImageForProposedRect:context:hints:` (page 1339)
  Returns a `CGImage` capturing the drawing of the receiver.

Managing Incremental Loads

- `cancelIncrementalLoad` (page 1339)
  Cancels the current download operation immediately, if the image is being incrementally loaded.

Image Accessibility

- `accessibilityDescription` (page 1334)
  Returns the image's accessibility description.
- `setAccessibilityDescription:` (page 1363)
  Sets the image's accessibility description.

Class Methods

canInitWithPasteboard:

Tests whether the receiver can create an instance of itself using pasteboard data.

```
+ (BOOL)canInitWithPasteboard:(NSPasteboard *)pasteboard
```

Parameters

pasteboard
  The pasteboard containing the image data.

Return Value
`YES` if the receiver knows how to handle the data on the pasteboard; otherwise, `NO`.

Discussion
This method uses the `NSImageRep` class method `imageUnfilteredPasteboardTypes` (page 1411) to find a class that can handle the data in the specified pasteboard. If you create your own `NSImageRep` subclasses, override the `imageUnfilteredPasteboardTypes` (page 1411) method to notify `NSImage` of the pasteboard types your class supports.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ imagePasteboardTypes` (page 1332)

Related Sample Code
CocoaDragAndDrop
People

Declared In
NSImage.h

imageFileTypes

Returns an array of strings identifying the image types supported by the registered NSImageRep objects.

```
+ (NSArray *)imageFileTypes
```

Return Value
An array of NSString objects, each of which identifies a single supported file type. The array can include encoded HFS file types as well as filename extensions.

Discussion
This list includes all file types supported by registered subclasses of NSImageRep plus those that can be converted to a supported type by a user-installed filter service. You can pass the array returned by this method directly to the runModalForTypes: (page 1821) method of NSOpenPanel.

When creating a subclass of NSImageRep, do not override this method. Instead, override the imageUnfilteredFileTypes (page 1410) method to notify NSImage of the file types your class supports directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ imageUnfilteredFileTypes (page 1333)

Related Sample Code
CocoaSlides
DeskPictAppDockMenu
Quartz Composer SlideShow
TrackBall

Declared In
NSImage.h

imageNamed:

Returns the NSImage instance associated with the specified name.

```
+ (id)imageNamed:(NSString *)name
```

Parameters
name
The name associated with the desired image.

Return Value
The NSImage object associated with the specified name, or nil if no such image was found.

Discussion
This method searches for named images in several places, returning the first image it finds matching the given name. The order of the search is as follows:

1. Search for an object whose name was set explicitly using the setName: method and currently resides in the image cache.
2. Search the application's main bundle for a file whose name matches the specified string. (For information on how the bundle is searched, see ""Accessing a Bundle's Contents"" in *Bundle Programming Guide*.)
3. Search the Application Kit framework for a shared image with the specified name.

When looking for files in the application bundle, it is better (but not required) to include the filename extension in the `name` parameter. When naming an image with the setName: method, it is also convention not to include filename extensions in the names you specify. That way, you can easily distinguish between images you have named explicitly and those you want to load from the application's bundle.

One particularly useful image you can retrieve is your application's icon. This image is set by Cocoa automatically and referenced by the string `@"NSApplicationIcon"`. Icons for other applications can be obtained through the use of methods declared in the `NSWorkspace` class. You can also retrieve many of the standard system images using Cocoa defined constants; for more information, see the "Image Template Constants" (page 1380), "Sharing Permissions Named Images" (page 1385), "System Entity Images" (page 1386), "Toolbar Named Images" (page 1387), and "View Type Template Images" (page 1390) sections for applicable constants.

If an application is linked in Mac OS X v10.5 or later, images requested using this method and whose name ends in the word "Template" are automatically marked as template images.

The `NSImage` class may cache a reference to the returned image object for performance in some cases. However, the class holds onto cached objects only while the object exists. If the image object is subsequently released, either because its retain count was 0 or it was not referenced anywhere in a garbage-collected application, the object may be quietly removed from the cache. Thus, if you plan to hold onto a returned image object, you must retain it like you would any Cocoa object. You can clear an image object from the cache explicitly by calling the object's setName: method and passing `nil` for the image name.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setName:` (page 1368)
`- name` (page 1360)
`iconForFile:` (page 3460) (NSWorkspace)
`+ imageFileTypes` (page 1330)

Related Sample Code
EnhancedDataBurn
FunHouse
GridCalendar
IconCollection
MenuMadness

Declared In
`NSImage.h`

imagePasteboardTypes

Returns an array of strings identifying the pasteboard types supported directly by the registered `NSImageRep` objects.

```
+ (NSArray *)imagePasteboardTypes
```

Return Value
An array of `NSString` objects, each of which identifies a single supported pasteboard type. By default, this list contains the `NSPDFPboardType`, `NSPICTPboardType`, `NSPostScriptPboardType`, and `NSTIFFPboardType` types.

Discussion
This list includes all pasteboard types supported by registered subclasses of `NSImageRep` plus those that can be converted to a supported type by a user-installed filter service.

When creating a subclass of `NSImageRep`, do not override this method. Instead, override the `imageUnfilteredPasteboardTypes` (page 1411) method to notify `NSImage` of the pasteboard types your class supports.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ imageUnfilteredPasteboardTypes` (page 1333)

Related Sample Code
CocoaDragAndDrop
GeekGameBoard
GLUT
Sketch+Accessibility
Sketch-112

Declared In
`NSImage.h`

imageTypes

Returns an array of UTI strings identifying the image types supported by the registered `NSImageRep` objects, either directly or through a user-installed filter service.

```
+ (NSArray *)imageTypes
```

Return Value
An array of `NSString` objects, each of which contains a UTI identifying a supported image type. Some sample image-related UTI strings include "`public.image`", "`public.jpeg`", and "`public.tiff`". For a list of supported types, see `UTCoreTypes.h`.

Discussion
The returned list includes UTIs all file types supported by registered subclasses of `NSImageRep` plus those that can be converted to a supported type by a user-installed filter service. You can use the returned UTI strings with any method that supports UTIs.

You should not override this method directly. Instead, you should override the `imageTypes` method of `NSImageRep`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ imageUnfilteredTypes` (page 1334)

Related Sample Code
AnimatedTableView
LightTable

Declared In
`NSImage.h`

imageUnfilteredFileTypes

Returns an array of strings identifying the file types supported directly by the registered `NSImageRep` objects.

```
+ (NSArray *)imageUnfilteredFileTypes
```

Return Value
An array of `NSString` objects, each of which identifies a single supported file type. File types are identified by file extension and HFS file types.

Discussion
The returned list does not contain pasteboard types that are available only through a user-installed filter service.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ imageFileTypes` (page 1330)

Declared In
`NSImage.h`

imageUnfilteredPasteboardTypes

Returns an array of strings identifying the pasteboard types supported directly by the registered `NSImageRep` objects.

```
+ (NSArray *)imageUnfilteredPasteboardTypes
```

Return Value
An array of `NSString` objects, each of which identifies a single supported pasteboard type.

Discussion
The returned list does not contain pasteboard types that are supported only through a user-installed filter service.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ imagePasteboardTypes` (page 1332)

Related Sample Code
GeekGameBoard

Declared In
`NSImage.h`

imageUnfilteredTypes

Returns an array of UTI strings identifying the image types supported directly by the registered `NSImageRep` objects.

```
+ (NSArray *)imageUnfilteredTypes
```

Return Value
An array of `NSString` objects, each of which contains a UTI identifying a supported image type. Some sample image-related UTI strings include "`public.image`", "`public.jpeg`", and "`public.tiff`". For a list of supported types, see `UTCoreTypes.h`.

Discussion
The returned list includes UTI strings only for those file types that are supported directly by registered subclasses of `NSImageRep`. It does not include types that are supported through user-installed filter services. You can use the returned UTI strings with any method that supports UTIs.

You should not override this method directly. Instead, you should override the `imageUnfilteredTypes` method of `NSImageRep`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ imageTypes` (page 1332)

Declared In
`NSImage.h`

Instance Methods

accessibilityDescription

Returns the image's accessibility description.

```
- (NSString *)accessibilityDescription
```

Return Value
A short localized string that does not include the name of the interface element.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setAccessibilityDescription:` (page 1363)

Declared In
`NSImage.h`

addRepresentation:

Adds the specified image representation object to to the receiver.

```
- (void)addRepresentation:(NSImageRep *)imageRep
```

Parameters

imageRep
The image representation to add.

Discussion
After invoking this method, you may need to explicitly set features of the new image representation, such as the size, number of colors, and so on. This is true particularly when the `NSImage` object has multiple image representations to choose from. See `NSImageRep` and its subclasses for the methods you use to complete initialization.

Any representation added by this method is retained by the receiver. Image representations cannot be shared among multiple `NSImage` objects.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `representations` (page 1362)
- `removeRepresentation:` (page 1362)

Related Sample Code
CocoaVideoFrameToNSImage
CompositeLab
Reducer
Son of Grab
StillMotion

Declared In
`NSImage.h`

addRepresentations:

Adds an array of image representation objects to the receiver.

```
- (void)addRepresentations:(NSArray *)imageReps
```

Parameters

imageReps

An array of `NSImageRep` objects.

Discussion

After invoking this method, you may need to explicitly set features of the new image representations, such as their size, number of colors, and so on. This is true particularly when the `NSImage` object has multiple image representations to choose from. See `NSImageRep` and its subclasses for the methods you use to complete initialization.

Representations added by this method are retained by the receiver. Image representations cannot be shared among multiple `NSImage` objects.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `representations` (page 1362)
- `removeRepresentation:` (page 1362)

Declared In

`NSImage.h`

alignmentRect

Returns alignment metadata that your code can use to position the image during layout.

```
- (NSRect)alignmentRect
```

Return Value

A rectangle containing the layout information for the image. If not set, the returned rectangle has an origin of (0, 0) and a size that matches the size of the image.

Discussion

The returned rectangle is merely a hint that your own code can use to determine positioning. The `NSImage` class does not use this rectangle during drawing. However, instances of `NSCell` typically use this information when laying out images within their own boundaries.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `setAlignmentRect:` (page 1363)

Declared In

`NSImage.h`

backgroundColor

Returns the background color of image.

```
- (NSColor *)backgroundColor
```

Return Value
The background color of the image. The default color is transparent, as returned by the `clearColor` (page 668) method of `NSColor`.

Discussion
The background color is visible only if the drawn image representation does not completely cover all of the pixels available for the image's current size.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSImage.h`

bestRepresentationForDevice:

Returns the best representation for the device with the specified characteristics. (Deprecated in Mac OS X v10.6.)

```
- (NSImageRep *)bestRepresentationForDevice:(NSDictionary *)deviceDescription
```

Parameters

deviceDescription
A dictionary of attributes for the specified device, or `nil` to specify the current device. For a list of dictionary keys and values appropriate to display and print devices, see the constants in `NSScreen`.

Return Value
The image representation that most closely matches the specified criteria.

Discussion
If *deviceDescription* is `nil`, this method uses the attributes of the device on which the content is to be drawn.

For information on how the "best" representation is chosen, see the “Images” chapter of *Cocoa Drawing Guide*.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `representations` (page 1362)
- `prefersColorMatch` (page 1361)
- `matchesOnMultipleResolution` (page 1360)
- `usesEPSOnResolutionMismatch` (page 1374)

Related Sample Code
LayerBackedOpenGLView
NSOpenGL Fullscreen
PDF Annotation Editor
Sketch-112

Declared In
NSImage.h

bestRepresentationForRect:context:hints:

Returns the best representation of the image for the specified rect using the provided hints.

```
- (NSImageRep *)bestRepresentationForRect:(NSRect)rect context:(NSGraphicsContext
    *)referenceContext hints:(NSDictionary *)hints
```

Parameters

rect
The area of the image to return.

referenceContext
A graphics context. This value can be nil.

hints
An optional dictionary of hints that provide more context for selecting or generating a CGImage, and may override properties of the *referenceContext*. See "Image Hint Dictionary Keys" (page 1375) for a summary of the possible key-value pairs.

Return Value
The image representation that most closely matches the specified criteria.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSImage.h

cacheDepthMatchesImageDepth

Returns a Boolean value indicating whether an image's offscreen window caches use the same bit depth as the image data itself. (Deprecated in Mac OS X v10.6. NSImage no longer caches to windows. A cache is now generated appropriate for the destination where an image is drawn. There is no replacement method.)

```
- (BOOL)cacheDepthMatchesImageDepth
```

Return Value
YES if the offscreen window caches use the same bit depth as the image data; otherwise, NO. The default value is NO.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- setCacheDepthMatchesImageDepth: (page 1364)

Declared In
NSImage.h

cacheMode

Returns the receiver's caching mode.

```
- (NSImageCacheMode)cacheMode
```

Return Value
A value indicating the caching mode. For a list of possible values, see `NSImageCacheMode` (page 1379). This value is set to `NSImageCacheDefault` by default.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- setCacheMode:` (page 1365)

Declared In
`NSImage.h`

cancelIncrementalLoad

Cancels the current download operation immediately, if the image is being incrementally loaded.

```
- (void)cancelIncrementalLoad
```

Discussion
This call has no effect if the image is not loading.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSImage.h`

CGImageForProposedRect:context:hints:

Returns a `CGImage` capturing the drawing of the receiver.

```
- (CGImageRef)CGImageForProposedRect:(NSRect *)proposedDestRect
    context:(NSGraphicsContext *)referenceContext hints:(NSDictionary *)hints
```

Parameters

proposedDestRect
On input, the proposed destination rectangle for drawing the image. If `NULL`, it defaults to the smallest pixel-integral rectangle containing {{0,0}, [self size]}. The *proposedDestRect* is in user space in the reference context.

referenceContext
A graphics context.

hints
A dictionary of hints that provide more context for selecting or generating a `CGImage`, and may override properties of the *referenceContext*.

Return Value
A `CGImageRef`. This may be an existing `CGImage` if one is available. If not, a new `CGImage` is created.

Discussion
An `NSImage` is potentially resolution independent, and may have representations that allow it to draw well in many contexts. A `CGImage` is more like a single pixel-based representation. This method produces a snapshot of how the `NSImage` would draw if it was asked to draw in the proposed rectangle in the graphics context.

All input parameters are optional. They provide hints for how to choose among existing CGImages, or how to create one if there isn't already a CGImage available. The parameters are only hints.

This method is typically called, not overridden.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSImage.h`

compositeToPoint:fromRect:operation:

Composites a portion of the image to the specified point in the current coordinate system. (Deprecated in Mac OS X v10.6.)

```
- (void)compositeToPoint:(NSPoint)aPoint fromRect:(NSRect)srcRect
    operation:(NSCompositingOperation)op
```

Parameters

aPoint
The point at which to draw the image, specified in the current coordinate system.

srcRect
The portion of the image you want to draw, specified in the image's coordinate system.

op
The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in "Constants" (page 1375).

Discussion
This method draws the specified portion of the image without checking the bounds rectangle you pass into the *srcRect* parameter. If you specify a source rectangle that strays outside of the image's bounds rectangle, it is conceivable that you could composite parts of the offscreen cache window that do not belong to the receiver's image. You can avoid this problem using the `drawAtPoint:fromRect:operation:fraction:` (page 1346) method, which checks the source rectangle before drawing.

During drawing, the image is composited from its offscreen window cache. Because the offscreen cache is not created until the image representation is first used, this method may need to render the image before compositing. Bitmap representations in particular are not cached until they are explicitly rendered. You can use the `lockFocus` (page 1358) and `unlockFocus` (page 1374) methods to force the cached version to be created.

Compositing part of an image is as efficient as compositing the whole image, but printing just part of an image is not. When printing, it's necessary to draw the whole image and rely on a clipping path to be sure that only the desired portion appears.

During printing, this method ignores the `op` parameter. Even though this parameter is ignored, this method attempts to render the image as close as possible to its appearance when the compositing operation is used on the screen. In either case, the best image representation is chosen for the printing context.

Important: If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the drawAtPoint:fromRect:operation:fraction: or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `dissolveToPoint:fromRect:fraction:` (page 1345)
- `drawAtPoint:fromRect:operation:fraction:` (page 1346)
- `drawInRect:fromRect:operation:fraction:` (page 1347)

Declared In
`NSImage.h`

compositeToPoint:fromRect:operation:fraction:

Composites a portion of the image at the specified opacity to the current coordinate system. (Deprecated in Mac OS X v10.6.)

```
- (void)compositeToPoint:(NSPoint)aPoint fromRect:(NSRect)srcRect
    operation:(NSCompositingOperation)op fraction:(CGFloat)delta
```

Parameters

aPoint
The point at which to draw the image, specified in the current coordinate system.

srcRect
The portion of the image you want to draw, specified in the image's coordinate system.

op
The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in "Constants" (page 1375).

delta
The desired opacity of the image, specified as a value between 0.0 and 1.0, with 1.0 representing total opacity. Values larger than 1.0 are interpreted as 1.0. This method always expects to render something, so for values that are equal to or less than 0, this method renders at full opacity.

Discussion
Behaves the same as `compositeToPoint:fromRect:operation:` (page 1340) except that you can specify the amount of opacity to use when drawing the image.

Important: If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the drawAtPoint:fromRect:operation:fraction: or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `dissolveToPoint:fromRect:fraction:` (page 1345)
- `drawAtPoint:fromRect:operation:fraction:` (page 1346)
- `drawInRect:fromRect:operation:fraction:` (page 1347)

Declared In
`NSImage.h`

compositeToPoint:operation:

Composites the entire image to the specified point in the current coordinate system. (Deprecated in Mac OS X v10.6.)

```
- (void)compositeToPoint:(NSPoint)aPoint operation:(NSCompositingOperation)op
```

Parameters

aPoint
The point at which to draw the image, specified in the current coordinate system.

op
The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in "Constants" (page 1375).

Discussion
This method draws the receiver's best image representation at the specified point in the currently focused view. The entire image is drawn using its current size information. During drawing, the image is composited from its offscreen window cache. Because the offscreen cache is not created until the image representation is first used, this method may need to render the image before compositing. Bitmap representations in particular are not cached until they are explicitly rendered. You can use the `lockFocus` (page 1358) and `unlockFocus` (page 1374) methods to force the cached version to be created.

During printing, this method ignores the `op` parameter. Even though this parameter is ignored, this method attempts to render the image as close as possible to its appearance when the compositing operation is used on the screen. In either case, the best image representation is chosen for the printing context.

Important: If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the drawAtPoint:fromRect:operation:fraction: or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

Availability
Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also
- `dissolveToPoint:fraction:` (page 1344)
- `drawAtPoint:fromRect:operation:fraction:` (page 1346)
- `drawInRect:fromRect:operation:fraction:` (page 1347)

Related Sample Code
Image Difference
QTKitMovieShuffler
RGB ValueTransformers
Sketch-112
STUCAuthoringDeviceCocoaSample

Declared In
`NSImage.h`

compositeToPoint:operation:fraction:

Composites the entire image at the specified opacity in the current coordinate system. (Deprecated in Mac OS X v10.6.)

```
- (void)compositeToPoint:(NSPoint)aPoint operation:(NSCompositingOperation)op
    fraction:(CGFloat)delta
```

Parameters

aPoint
The point at which to draw the image, specified in the current coordinate system.

op
The compositing operation to use when drawing the image to the screen. The supported compositing operations are described in "Constants" (page 1375).

delta
The desired opacity of the image, specified as a value between 0.0 and 1.0, with 1.0 representing total opacity. Values larger than 1.0 are interpreted as 1.0. This method always expects to render something, so for values that are equal to or less than 0, this method renders at full opacity.

Discussion
Behaves the same as `compositeToPoint:operation:` (page 1342) except that you can specify the amount of opacity to use when drawing the image.

Important: If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the drawAtPoint:fromRect:operation:fraction: or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `dissolveToPoint:fraction:` (page 1344)
- `drawAtPoint:fromRect:operation:fraction:` (page 1346)
- `drawInRect:fromRect:operation:fraction:` (page 1347)

Declared In
`NSImage.h`

delegate

Returns the delegate object of the receiver

```
- (id < NSImageDelegate >)delegate
```

Return Value
The current delegate object, or `nil` if no delegate has been set.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDelegate:` (page 1366)

Declared In
`NSImage.h`

dissolveToPoint:fraction:

Composites the entire image to the specified location using the `NSCompositeSourceOver` operator. (Deprecated in Mac OS X v10.6.)

```
- (void)dissolveToPoint:(NSPoint)aPoint fraction:(CGFloat)delta
```

Parameters

aPoint
The point at which to draw the image, specified in the current coordinate system.

delta
The desired opacity of the image, specified as a value between 0.0 and 1.0. A value of 0.0 renders the image totally transparent while 1.0 renders it fully opaque. Values larger than 1.0 are interpreted as 1.0.

Discussion
Except for the choice of compositing operator, this method behaves in the same way as the `compositeToPoint:operation:` (page 1342) method. During printing, the *delta* parameter is ignored.

If the source image contains alpha information, this operation may promote the destination `NSWindow` object to contain alpha information.

To slowly dissolve this image onto another, you can invoke this method (or the `dissolveToPoint:fromRect:fraction:` (page 1345) method) repeatedly with an ever-increasing *delta* value. Because the *delta* parameter refers to the visible fraction of the source image, increasing the value causes the source image to replace the destination content gradually. You should generally perform this type of operation using a buffered window or other offscreen drawing environment.

Special Considerations

If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the drawAtPoint:fromRect:operation:fraction: or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

Availability
Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

Declared In
`NSImage.h`

dissolveToPoint:fromRect:fraction:

Composites a portion of the image to the specified location using the `NSCompositeSourceOver` operator. (Deprecated in Mac OS X v10.6.)

```
- (void)dissolveToPoint:(NSPoint)aPoint fromRect:(NSRect)srcRect
    fraction:(CGFloat)delta
```

Parameters

aPoint

The point at which to draw the image, specified in the current coordinate system.

srcRect

The portion of the image you want to draw, specified in the image's coordinate system.

delta

The desired opacity of the image, specified as a value between 0.0 and 1.0. A value of 0.0 renders the image totally transparent while 1.0 renders it fully opaque. Values larger than 1.0 are interpreted as 1.0.

Discussion
Except for the choice of compositing operator, this method behaves in the same way as the `compositeToPoint:fromRect:operation:` (page 1340) method. During printing, the *delta* parameter is ignored.

If the source image contains alpha information, this operation may promote the destination `NSWindow` object to contain alpha information.

Special Considerations

If you are writing new code, or updating old code, you should avoid using this method. Instead, you should use the drawAtPoint:fromRect:operation:fraction: or `drawInRect:fromRect:operation:fraction:` method to draw the image. Although the method itself is not deprecated, the behavior it provides is not recommended for general use.

Availability
Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also
- `dissolveToPoint:fraction:` (page 1344)

Declared In
`NSImage.h`

drawAtPoint:fromRect:operation:fraction:

Draws all or part of the image at the specified point in the current coordinate system.

```
- (void)drawAtPoint:(NSPoint)point fromRect:(NSRect)srcRect
    operation:(NSCompositingOperation)op fraction:(CGFloat)delta
```

Parameters

point

The location in the current coordinate system at which to draw the image.

srcRect

The source rectangle specifying the portion of the image you want to draw. The coordinates of this rectangle are specified in the image's own coordinate system. If you pass in `NSZeroRect`, the entire image is drawn.

op

The compositing operation to use when drawing the image. See the `NSCompositingOperation` (page 1375) constants.

delta

The opacity of the image, specified as a value from 0.0 to 1.0. Specifying a value of 0.0 draws the image as fully transparent while a value of 1.0 draws the image as fully opaque. Values greater than 1.0 are interpreted as 1.0.

Discussion
The image content is drawn at its current resolution and is not scaled unless the CTM of the current coordinate system itself contains a scaling factor. The image is otherwise positioned and oriented using the current coordinate system.

Unlike the `compositeToPoint:fromRect:operation:` (page 1340) and `compositeToPoint:fromRect:operation:fraction:` (page 1341) methods, this method checks the rectangle you pass to the *srcRect* parameter and makes sure it does not lie outside the image bounds.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `dissolveToPoint:fraction:` (page 1344)
- `drawInRect:fromRect:operation:fraction:` (page 1347)

Related Sample Code
ComplexBrowser
Reducer

Declared In
`NSImage.h`

drawInRect:fromRect:operation:fraction:

Draws all or part of the image in the specified rectangle in the current coordinate system.

```
- (void)drawInRect:(NSRect)dstRect fromRect:(NSRect)srcRect
    operation:(NSCompositingOperation)op fraction:(CGFloat)delta
```

Parameters

dstRect

The rectangle in which to draw the image, specified in the current coordinate system.

srcRect

The source rectangle specifying the portion of the image you want to draw. The coordinates of this rectangle must be specified using the image's own coordinate system. If you pass in `NSZeroRect`, the entire image is drawn.

op

The compositing operation to use when drawing the image. See the `NSCompositingOperation` (page 1375) constants.

delta

The opacity of the image, specified as a value from 0.0 to 1.0. Specifying a value of 0.0 draws the image as fully transparent while a value of 1.0 draws the image as fully opaque. Values greater than 1.0 are interpreted as 1.0.

Discussion

If the `srcRect` and `dstRect` rectangles have different sizes, the source portion of the image is scaled to fit the specified destination rectangle. The image is otherwise positioned and oriented using the current coordinate system.

Unlike the `compositeToPoint:fromRect:operation:` (page 1340) and `compositeToPoint:fromRect:operation:fraction:` (page 1341) methods, this method checks the rectangle you pass to the *srcRect* parameter and makes sure it does not lie outside the image bounds.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- dissolveToPoint:fraction:` (page 1344)

`- drawAtPoint:fromRect:operation:fraction:` (page 1346)

Related Sample Code

AnimatedTableView

CocoaVideoFrameToNSImage

PhotoSearch

Transformed Image

WebKitDOMElementPlugIn

Declared In

`NSImage.h`

drawInRect:fromRect:operation:fraction:respectFlipped:hints:

Draws all or part of the image in the specified rectangle respecting the flippedness and hints.

```
- (void)drawInRect:(NSRect)dstSpacePortionRect fromRect:(NSRect)srcSpacePortionRect
    operation:(NSCompositingOperation)op fraction:(CGFloat)requestedAlpha
   respectFlipped:(BOOL)respectContextIsFlipped hints:(NSDictionary *)hints
```

Parameters

dstSpacePortionRect

The rectangle in which to draw the image, specified in the current coordinate system.

srcSpacePortionRect

The source rectangle specifying the portion of the image you want to draw. The coordinates of this rectangle must be specified using the image's own coordinate system. If you pass in `NSZeroRect`, the entire image is drawn.

op

The compositing operation to use when drawing the image. See the `NSCompositingOperation` (page 1375) constants.

requestedAlpha

The alpha of the image, specified as a value from 0.0 to 1.0. Specifying a value of 0.0 draws the image as fully transparent while a value of 1.0 draws the image as fully opaque. Values greater than 1.0 are interpreted as 1.0.

respectContextIsFlipped

`YES` if the drawing should respect the context flipped state, otherwise `NO`.

hints

An optional dictionary of hints that provide more context for selecting or generating the image. See "Image Hint Dictionary Keys" (page 1375) for a summary of the possible key-value pairs.

Discussion

If the *srcSpacePortionRect* and *dstSpacePortionRect* rectangles have different sizes, the source portion of the image is scaled to fit the specified destination rectangle.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSImage.h`

drawRepresentation:inRect:

Draws the image using the specified image representation object.

```
- (BOOL)drawRepresentation:(NSImageRep *)imageRep inRect:(NSRect)dstRect
```

Parameters

imageRep

The image representation object to be drawn.

dstRect

The rectangle in which to draw the image representation, specified in the current coordinate system.

Return Value

`YES` if the image was successfully drawn; otherwise, returns `NO`.

Discussion
This method fills the specified rectangle with the image's current background color and then sends a message to the specified image representation asking if to draw itself. If the image supports the ability to scale itself when it is resized, this method sends a `drawInRect:` (page 1417) message; otherwise, it sends a `drawAtPoint:` (page 1416) message.

You should not call this method directly; an `NSImage` object uses it to cache and print its image representations. You can override this method to change the way images are rendered into their caches and onto the printed page. For example, you could scale or rotate the coordinate system before sending this message to `super` to continue rendering the image representation.

If the background color is fully transparent and the image data is not being cached, the specified rectangle is not to be filled before the representation draws.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- backgroundColor` (page 1336)

Declared In
`NSImage.h`

hitTestRect:withImageDestinationRect:context:hints:flipped:

Returns whether the destination rectangle would intersect a non-transparent portion of the image.

```
- (BOOL)hitTestRect:(NSRect)testRectDestSpace withImageDestinationRect:(NSRect)imageRectDestSpace context:(NSGraphicsContext *)referenceContext hints:(NSDictionary *)hints flipped:(BOOL)flipped
```

Parameters

testRectDestSpace
The rectangle to hit test.

imageRectDestSpace
A rectangle representing the drawn size of the image.

referenceContext
A graphics context. This value can be `nil`.

hints
An optional dictionary of hints that provide more context for selecting or generating a `CGImage`, and may override properties of the *referenceContext*. See "Image Hint Dictionary Keys" (page 1375) for a summary of the possible key-value pairs.

flipped
`YES` if the image is flipped, otherwise `NO`.

Return Value
YES if the *testRectDestSpace* intersects with non-transparent content within the *imageRectDestSpace*, otherwise NO.

Discussion
This method simulates the results of hit-testing the test rectangle as if the image was drawn in the graphics context using the provided hints and respecting the specified flippedness..

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSImage.h`

initByReferencingFile:

Initializes and returns an `NSImage` instance and associates it with the specified file.

```
- (id)initByReferencingFile:(NSString *)filename
```

Parameters

filename

A full or relative path name specifying the file with the desired image data. Relative paths must be relative to the current working directory.

Return Value
An initialized `NSImage` instance, or `nil` if the new instance cannot be initialized.

Discussion
This method initializes the image object lazily. It does not actually open the specified file or create any image representations from its data until an application attempts to draw the image or request information about it.

The *filename* parameter should include the file extension that identifies the type of the image data. The mechanism that actually creates the image representation for *filename* looks for an `NSImageRep` subclass that handles that data type from among those registered with `NSImage`.

Because this method doesn't actually create image representations for the image data, your application should do error checking before attempting to use the image; one way to do so is by invoking the `isValid` (page 1357) method to check whether the image can be drawn.

This method invokes `setDataRetained:` (page 1366) with an argument of `YES`, thus enabling it to hold onto its filename. When archiving an image created with this method, only the image's filename is written to the archive.

If the cached version of the image uses less memory than the original image data, the original data is flushed and the cached image is used. (This can occur for images whose resolution is greater than 72 dpi.) If you resize the image by less than 50%, the data is loaded in again from the file. If you expect the file to change or be deleted, you should use `initWithContentsOfFile:` (page 1352) instead.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CompositeLab
FilterDemo
ImagePicker
PictureTaker
Rulers

Declared In
`NSImage.h`

initByReferencingURL:

Initializes and returns an `NSImage` instance and associates it with the specified URL.

```
- (id)initByReferencingURL:(NSURL *)url
```

Parameters

url

The URL identifying the image.

Return Value

An initialized `NSImage` instance, or `nil` if the new instance cannot be initialized.

Discussion

This method initializes the image object lazily. It does not attempt to retrieve the data from the specified URL or create any image representations from that data until an application attempts to draw the image or request information about it.

This `url` parameter should include a file extension that identifies the type of the image data. The mechanism that actually creates the image representation looks for an `NSImageRep` subclass that handles that data type from among those registered with `NSImage`.

Because this method doesn't actually create image representations for the image data, your application should do error checking before attempting to use the image; one way to do so is by invoking the `isValid` (page 1357) method to check whether the image can be drawn.

This method invokes `setDataRetained:` (page 1366) with an argument of `YES`, thus enabling it to hold onto its URL. When archiving an image created with this method, only the image's URL is written to the archive.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

AnimatedTableView

TrackBall

Declared In

`NSImage.h`

initWithCGImage:size:

Initializes and returns an `NSImage` instance with the contents of the CGImage.

```
- (id)initWithCGImage:(CGImageRef)cgImage size:(NSSize)size
```

Parameters

cgImage

The source `CGImage`.

size

The size of the new image. If size is `NSZeroSize`, the pixel dimensions of *cgImage* are assumed as the image's size.

Return Value

An initialized `NSImage` instance, or `nil` if the new instance cannot be initialized.

Discussion
You should not assume anything about the image, other than that drawing it is equivalent to drawing the `CGImage`.

This is not a designated initializer.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
CameraBrowser
QuickLookDownloader
SimpleCameraBrowser

Declared In
`NSImage.h`

initWithContentsOfFile:

Initializes and returns an `NSImage` instance with the contents of the specified file.

```
- (id)initWithContentsOfFile:(NSString *)filename
```

Parameters

filename

A full or relative path name specifying the file with the desired image data. Relative paths must be relative to the current working directory.

Return Value
An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the specified file.

Discussion
Unlike `initByReferencingFile:` (page 1350), which initializes an `NSImage` object lazily, this method immediately opens the specified file and creates one or more image representations from its data.

The *filename* parameter should include the file extension that identifies the type of the image data. This method looks for an `NSImageRep` subclass that handles that data type from among those registered with `NSImage`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSImage.h`

initWithContentsOfURL:

Initializes and returns an `NSImage` instance with the contents of the specified URL.

```
- (id)initWithContentsOfURL:(NSURL *)aURL
```

Parameters

aUrl

The URL identifying the image.

Return Value

An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the specified URL.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CocoaCreateMovie

Declared In

`NSImage.h`

initWithData:

Initializes and returns an `NSImage` instance with the contents of the specified `NSData` object.

```
- (id)initWithData:(NSData *)data
```

Parameters

data

The data object containing the image data.

Return Value

An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the specified data object.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSImage.h`

initWithDataIgnoringOrientation:

Initializes and returns an `NSImage` instance with the contents of the specified `NSData` object, ignoring the EXIF orientation tags..

```
- (id)initWithDataIgnoringOrientation:(NSData *)data
```

Parameters

data

The data object containing the image data.

Return Value

An initialized `NSImage` instance, or `nil` if the method cannot create an image representation from the contents of the specified data object.

Availability

Available in Mac OS X v10.6 and later.

Declared In
NSImage.h

initWithIconRef:

Initializes the image object with a Carbon-style icon resource.

```
- (id)initWithIconRef:(IconRef)iconRef
```

Parameters

iconRef

A reference to a Carbon icon resource.

Return Value
An initialized NSImage instance.

Discussion
Creates one or more bitmap image representations, one for each size icon contained in the IconRef data structure. This initialization method automatically retains the data in the iconRef parameter and loads the bitmaps from that data file lazily.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSImage.h

initWithPasteboard:

Initializes and returns an NSImage instance with data from the specified pasteboard.

```
- (id)initWithPasteboard:(NSPasteboard *)pasteboard
```

Parameters

pasteboard

The pasteboard containing the image data.

Return Value
An initialized NSImage instance, or nil if the method cannot create an image representation from the contents of the pasteboard.

Discussion
The specified pasteboard should contain a type supported by one of the registered NSImageRep subclasses. Table 61-1 lists the default pasteboard types and file extensions for several NSImageRep subclasses.

Table 61-1 Default pasteboard types for image representations

Image representation class	Default pasteboard type	Default file extensions
NSBitmapImageRep	NSTIFFPboardType	tiff, gif, jpg, and others
NSPDFImageRep	NSPDFPboardType	pdf
NSEPSImageRep	NSPostscriptPboardType	eps

Image representation class	Default pasteboard type	Default file extensions
NSPICTImageRep	NSPICTPboardType	pict

If the specified pasteboard contains the value NSFilenamesPboardType, each filename on the pasteboard should have an extension supported by one of the registered NSImageRep subclasses. You can use the imageUnfilteredFileTypes (page 1410) method of a given subclass to obtain the list of supported types for that class.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSImage.h

initWithSize:

Initializes and returns an NSImage instance whose size is set to the specified value.

```
- (id)initWithSize:(NSSize)aSize
```

Parameters

aSize

The size of the image, measured in points.

Return Value
An initialized NSImage instance.

Discussion
This method does not add any image representations to the image object.. It is permissible to initialize the receiver by passing a size of (0.0, 0.0); however, the receiver's size must be set to a non-zero value before the NSImage object is used or an exception will be raised.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setSize: (page 1370)

Related Sample Code
CompositeLab
FunHouse
RGB Image
RGB ValueTransformers
Sketch-112

Declared In
NSImage.h

isCachedSeparately

Returns a Boolean value indicating whether each image representation caches its contents in a separate offscreen window. (Deprecated in Mac OS X v10.6. NSImage no longer caches to windows. There is no replacement method)

```
- (BOOL)isCachedSeparately
```

Return Value
`YES` if the image representations cache their content in separate offscreen windows; otherwise, `NO`. The default value is `NO`.

Discussion
If this method returns `NO`, it means that the image may be cached in a shared window but is not required to be. Images are cached in a shared window if they have the same general attributes, such as color space, resolution, and bit depth.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

Declared In
`NSImage.h`

isDataRetained

Returns a Boolean value indicating whether the receiver retains its source image data. (Deprecated in Mac OS X v10.6. In Mac OS v10.6, `NSImage` no longer discards data in such a way that the original can no longer be reconstructed. There is no replacement method.)

```
- (BOOL)isDataRetained
```

Return Value
`YES` if the image retains its source data; otherwise, `NO`. The default value is `NO` with some exceptions, which are covered in the discussion.

Discussion
For image objects initialized using either the `initByReferencingFile:` (page 1350) or `initByReferencingURL:` (page 1351) method, this value is `YES` by default. The reason is that for these methods, data retention simply involves retaining the filename or URL.

Data retention increases the memory used by the `NSImage` object and its image representations.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

Declared In
`NSImage.h`

isFlipped

Returns a Boolean value indicating whether the image uses a flipped coordinate system. (Deprecated in Mac OS X v10.6. The flipped property of an image was widely misunderstood and has been deprecated. Use `drawInRect:fromRect:operation:fraction:respectFlipped:hints:` (page 1347) to draw respecting a context's flipped status and `lockFocusFlipped:` (page 1359) to draw into a flipped image.)

```
- (BOOL)isFlipped
```

Return Value
`YES` if the image's coordinate system is flipped; otherwise, `NO`. The default is `NO`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
`- setFlipped:` (page 1367)

Declared In
`NSImage.h`

isTemplate

Returns a Boolean value indicating whether the image is a template image.

```
- (BOOL)isTemplate
```

Return Value
`YES` if the image is a template image; otherwise, `NO`.

Discussion
Template images consist of black and clear colors (and an alpha channel). Template images are not intended to be used as standalone images and are usually mixed with other content to create the desired final appearance.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setTemplate:` (page 1371)

Declared In
`NSImage.h`

isValid

Returns a Boolean value indicating whether an image representation from the receiver can be drawn.

```
- (BOOL)isValid
```

Return Value
`YES` if the receiver can be drawn; otherwise, `NO`.

Discussion
If the receiver is initialized with an existing image file, but the corresponding image data is not yet loaded into memory, this method loads the data and expands it as needed. If the receiver contains no image representations and no associated image file, this method creates a valid cached image representation and initializes it to the default bit depth. This method returns `NO` in cases where the file or URL from which it was initialized is nonexistent or when the data in an existing file is invalid.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initByReferencingFile:` (page 1350)
- `initByReferencingURL:` (page 1351)

Declared In
`NSImage.h`

lockFocus

Prepares the image to receive drawing commands.

```
- (void)lockFocus
```

Discussion
This method sets the current drawing context to the area of the offscreen window used to cache the receiver's contents. Subsequent drawing commands are composited to this offscreen window. If the offscreen drawing area already has some content, any new drawing commands are composited with that content. This method does not modify the original image data directly.

When locking focus, this method chooses the best image representation object available and locks focus on that object. If the receiver has no image representations, this method creates one with the default depth and locks focus on it. For information on how the "best" representation is chosen, see the “Images” chapter of *Cocoa Drawing Guide*.

A successful `lockFocus` message must be balanced with a matching `unlockFocus` (page 1374) message to the same `NSImage` object. These messages bracket the code that draws the image.

If `lockFocus` is unable to focus on the image, it raises an `NSImageCacheException`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `bestRepresentationForDevice:` (page 1337)
- `isValid` (page 1357)
- `prefersColorMatch` (page 1361)
- `representations` (page 1362)

Related Sample Code
FunHouse
Image Difference
RGB Image
RGB ValueTransformers

Sketch-112

Declared In
NSImage.h

lockFocusFlipped:

Prepares the image to receive drawing commands using the specified flipped state.

```
- (void)lockFocusFlipped:(BOOL)flipped
```

Parameters

flipped

YES if the drawing context should be flipped, otherwise NO.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSImage.h

lockFocusOnRepresentation:

Prepares the specified image representation to receive drawing commands. (Deprecated in Mac OS X v10.6. Use the code fragment shown in the special considerations below.)

```
- (void)lockFocusOnRepresentation:(NSImageRep *)imageRepresentation
```

Parameters

imageRepresentation

An image representation belonging to the receiver, or nil if you want the receiver to choose which image representation to use.

Discussion
This method sets the current drawing context to the area of the offscreen window used to cache the specified image representation's contents. Subsequent drawing commands are composited to this offscreen window. If the offscreen drawing area already has some content, any new drawing commands are composited with that content. This method does not modify the original image data directly.

If *imageRepresentation* is nil, this method acts like the lockFocus (page 1358) method, setting the focus to the best representation for the NSImage object.

A successful lockFocusOnRepresentation: message must be balanced with a matching unlockFocus (page 1374) message to the same NSImage object. These messages bracket the code that draws the image.

If lockFocusOnRepresentation: is unable to focus on the specified image representation, it raises an NSImageCacheException.

Special Considerations

This method is deprecated as it did not set up *imageRepresentation* as a drawing destination, it set the image up as a drawing destination, then drew *imageRepresentation* into it. You can replace this functionality with the following code fragment

```
[image lockFocus];
[imageRepresentation drawInRect:NSMakeRect(0,0,[image size].width, [image
size].height)];

image unlockFocus;
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `isValid` (page 1357)

Declared In
`NSImage.h`

matchesOnMultipleResolution

Returns a Boolean value indicating whether image representations whose resolution is an integral multiple of the device resolution are considered a match.

```
- (BOOL)matchesOnMultipleResolution
```

Return Value
`YES` if image representations whose resolution is an integral multiple of the device resolution are considered a match; otherwise, `NO`.

Discussion
When this method returns `NO`, only image representations whose resolution is exactly the same as the device resolution are considered matches. If this method returns `YES` and multiple image representations fit this criteria, the one whose resolution is closest to the device resolution is chosen.

The default value is `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMatchesOnMultipleResolution:` (page 1368)

Declared In
`NSImage.h`

name

Returns the name associated with the receiver, if any.

```
- (NSString *)name
```

Return Value
The name associated with the receiver, or `nil` if no name is assigned.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setName:` (page 1368)

Declared In
`NSImage.h`

prefersColorMatch

Returns a Boolean value indicating whether the image prefers to choose image representations using color matching or resolution matching.

```
- (BOOL)prefersColorMatch
```

Return Value
`YES` if color matching is preferred over resolution matching; otherwise `NO` if resolution matching is preferred.

Discussion
Both color matching and resolution matching may influence the choice of an image representation. This method simply indicates which technique is used first during the selection process. The default value is `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPrefersColorMatch:` (page 1369)

Declared In
`NSImage.h`

recache

Invalidates and frees the offscreen caches of all image representations.

```
- (void)recache
```

Discussion
If you modify an image representation, you must send a `recache` (page 1361) message to the corresponding image object to force the changes to be recached. The next time any image representation is drawn, it is asked to recreate its cached image. If you do not send this message, the image representation may use the old cache data. This method simply clears the cached image data; it does not delete the `NSCachedImageRep` objects associated with any image representations.

If you do not plan to use an image again right away, you can free its caches to reduce the amount of memory consumed by your program.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
TextInputView

Declared In
`NSImage.h`

removeRepresentation:

Removes the specified image representation from the receiver and releases it.

```
- (void)removeRepresentation:(NSImageRep *)imageRep
```

Parameters

imageRep

The image representation object you want to remove.

Availability
Available in Mac OS X v10.0 and later.

See Also
- representations (page 1362)

Declared In
NSImage.h

representations

Returns an array containing all of the receiver's image representations.

```
- (NSArray *)representations
```

Return Value
An array containing zero or more NSImageRep objects.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaCreateMovie
Reducer

Declared In
NSImage.h

scalesWhenResized

Returns a Boolean value indicating whether image representations are scaled to fit the receiver's size. (Deprecated in Mac OS X v10.6. This method was related to caching behavior. In Mac OS X v10.6 and later image caching is no long necessary and as a resuilt there is no replacement necessary.)

```
- (BOOL)scalesWhenResized
```

Return Value
YES if image representations are scaled to fit the receiver; otherwise, NO. The default value is NO.

Discussion
Images are not resized during drawing if this method returns YES. They are only resized when you change the size by sending the receiver a setSize: (page 1370) message.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `setScalesWhenResized:` (page 1369)

Declared In
`NSImage.h`

setAccessibilityDescription:

Sets the image's accessibility description.

```
- (void)setAccessibilityDescription:(NSString *)description
```

Parameters
description
A short localized string that does not include the name of the interface element.

Discussion
This description will be used automatically by interface elements that display images. Like all accessibility descriptions, the string should be a short localized string that does not include the name of the interface element. For instance, "delete" rather than "delete button".

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSImage.h`

setAlignmentRect:

Sets the alignment metadata that your code can use to position the image during layout.

```
- (void)setAlignmentRect:(NSRect)rect
```

Parameters
rect
The alignment rectangle for the image.

Discussion
Alignment rectangles specify baselines that you can use to position the content of an image more accurately. These baselines are merely hints that your own code can use to determine positioning and are not used internally by `NSImage` itself during drawing. For example, if you have a 20 x 20 pixel icon that includes a glow effect, you might set the alignment rectangle to {{2, 2}, {16, 16}} to indicate the position of the underlying icon without the glow effect.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `alignmentRect` (page 1336)

Declared In
NSImage.h

setBackgroundColor:

Sets the background color of the image.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Parameters

aColor

The new background color for the image.

Discussion
The background color is visible only if the drawn image representation does not completely cover all of the pixels available for the image's current size. The background color is ignored for cached image representations; such caches are always created with a white background. This method does not cause the receiver to recache itself.

Availability
Available in Mac OS X v10.0 and later.

See Also
- recache (page 1361)
- backgroundColor (page 1336)

Declared In
NSImage.h

setCacheDepthMatchesImageDepth:

Sets whether the receiver's offscreen window caches use the same bit depth as the image data itself. (Deprecated in Mac OS X v10.6. NSImage no longer caches to windows. A cache is now generated appropriate for the destination where an image is drawn. There is no replacement method.)

```
- (void)setCacheDepthMatchesImageDepth:(BOOL)flag
```

Parameters

flag

YES if the offscreen caches use the same bit-depth associated with the image data; otherwise, NO to indicate they should use the default bit depth.

Discussion
This method does not cause the receiver to recache itself. The default depth limit is equal to the bit depth of the deepest screen on the system.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- cacheDepthMatchesImageDepth (page 1338)
- lockFocus (page 1358)

- recache (page 1361)

Declared In
NSImage.h

setCachedSeparately:

Sets whether each image representation uses a separate offscreen window to cache its contents. (Deprecated in Mac OS X v10.6. NSImage no longer caches to windows. There is no replacement method)

```
- (void)setCachedSeparately:(BOOL)flag
```

Parameters

flag

YES if you want each of the receiver's image representation objects to use a separate offscreen window for caching; otherwise, NO.

Discussion
If you specify NO, a representation can be cached together with other images, though in practice it might not be. This method does not invalidate any existing caches.

If you plan to resize an NSImage object frequently, it is usually more efficient to cache its representations separately. In some situations, you might also want to enable separate caching if you plan to use the compositeToPoint:fromRect:operation: (page 1340) or compositeToPoint:fromRect:operation:fraction: (page 1341)methods to draw the image.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- recache (page 1361)

Declared In
NSImage.h

setCacheMode:

Set the receiver's caching mode.

```
- (void)setCacheMode:(NSImageCacheMode)mode
```

Parameters

mode

The caching mode to use with this image. For a list of possible values, see NSImageCacheMode (page 1379).

Discussion
The caching mode determines when the receiver's image representations use offscreen caches. Offscreen caches speed up rendering time but do so by using extra memory. In the default caching mode (NSImageCacheDefault), each image representation chooses the caching technique that produces the

fastest drawing times. For example, in the default mode, the `NSPDFImageRep` and `NSEPSImageRep` classes use the `NSImageCacheAlways` mode but the `NSBitmapImageRep` class uses the `NSImageCacheBySize` mode.

For more information on image caching behavior, see the "Images" chapter of *Cocoa Drawing Guide*.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- cacheMode` (page 1339)

Declared In
`NSImage.h`

setDataRetained:

Sets whether the receiver retains its source image data. (Deprecated in Mac OS X v10.6. In Mac OS v10.6, `NSImage` no longer discards data in such a way that the original can no longer be reconstructed. There is no replacement method.)

```
- (void)setDataRetained:(BOOL)flag
```

Parameters

flag

`YES` if you want the source image data to be retained; otherwise `NO`.

Discussion
Retention of the source image data is important if the source of the image data could change, be moved, or be deleted. Data retention is also useful if you plan to resize an image frequently; otherwise, resizing occurs on a cached copy of the image, which can lose image quality during successive scaling operations. With data retention enabled, the image is resized from the original source data.

If the responsibility for drawing the image is delegated to another object, there is no reason to retain the image data. Similarly, if the source of the image data is not expected to change or you do not plan to resize the image, you do not need to retain the data. In fact, retaining the data leads to increased memory usage, which could have a negative impact on performance.

If you create your image object using the `initByReferencingFile:` (page 1350) method, the only data retained is the name of the source file.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

Declared In
`NSImage.h`

setDelegate:

Sets the delegate object of the receiver.

```
- (void)setDelegate:(id < NSImageDelegate >)anObject
```

Parameters

anObject

The new delegate object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `delegate` (page 1344)

Declared In

`NSImage.h`

setFlipped:

Sets whether the polarity of the y axis is inverted when drawing an image. (Deprecated in Mac OS X v10.6. The flipped property of an image was widely misunderstood and has been deprecated. Use `drawInRect:fromRect:operation:fraction:respectFlipped:hints:` (page 1347) to draw respecting a context's flipped status and `lockFocusFlipped:` (page 1359) to draw into a flipped image.)

```
- (void)setFlipped:(BOOL)flag
```

Parameters

flag

`YES` if you want the image data to be inverted before drawing; otherwise, `NO`.

Discussion

If *flag* is `YES`, the y-axis of the image's internal coordinate system is inverted, with the origin in the upper-left corner and the positive y axis extending downward. This method affects only the coordinate system used internally by the image and the orientation of the image when it is drawn; it does not affect the coordinate system used to specify the position of an image in a view. This method does not cause the receiver to recache itself.

If you set *flag* to `YES` and then lock focus and draw into the image, the content you draw is cached in the inverted (flipped) orientation. Changing the value for *flag* does not affect the orientation of the cached image.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- `isFlipped` (page 1357)
- `recache` (page 1361)

Related Sample Code

ImageMap

ImageMapExample

PhotoSearch

Sketch-112

WebKitPluginStarter

Declared In
NSImage.h

setMatchesOnMultipleResolution:

Sets whether image representations whose resolutions are integral multiples of the device resolution are considered a match.

```
- (void)setMatchesOnMultipleResolution:(BOOL)flag
```

Parameters

flag

YES if image representations whose resolution is an integral multiple of the device resolution should be considered a match; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- matchesOnMultipleResolution (page 1360)

Declared In
NSImage.h

setName:

Registers the receiver under the specified name.

```
- (BOOL)setName:(NSString *)aString
```

Parameters

aString

The name to associate with the receiver.

Return Value
YES if the receiver was successfully registered with the given name; otherwise, NO.

Discussion
If the receiver is already registered under a different name, this method unregisters the other name. If a different image is registered under the name specified in *aString*, this method does nothing and returns NO.

When naming an image using this method, it is convention not to include filename extensions in the names you specify. That way, you can easily distinguish between images you have named explicitly and those you want to load from the application's bundle. For information about the rules used to search for images, and for information about the ownership policy of named images, see the imageNamed: method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- name (page 1360)
+ imageNamed: (page 1330)

Related Sample Code
ClockControl
QTKitMovieShuffler

Declared In
NSImage.h

setPrefersColorMatch:

Sets whether choosing an image representation favors color matching over resolution matching.

```
- (void)setPrefersColorMatch:(BOOL)flag
```

Parameters

flag

YES if the receiver should match the color capabilities of the rendering device first; otherwise, NO to indicate that resolution matching is preferred.

Discussion
Both color matching and resolution matching may influence the choice of an image representation. You use this method to choose which technique should be used first during the selection process.

Availability
Available in Mac OS X v10.0 and later.

See Also
- prefersColorMatch (page 1361)

Declared In
NSImage.h

setScalesWhenResized:

Sets whether different-sized image representations are scaled to fit the receiver's size. (Deprecated in Mac OS X v10.6. This method was related to caching behavior. In Mac OS X v10.6 and later image caching is no long necessary and as a resuilt there is no replacement necessary.)

```
- (void)setScalesWhenResized:(BOOL)flag
```

Parameters

flag

YES if image representations are scaled to fit; otherwise NO.

Discussion
Most images (especially those loaded from files and URLs) contain only a single image representation whose size is the same as the receiver. It is possible to add image representations using the addRepresentation: (page 1335) or addRepresentations: (page 1335) methods but doing so is rarely necessary because modern hardware is powerful enough to resize and scale images quickly. The only reason to consider creating new representations is if each representations contains a customized version of the image at a specific size. (TIFF images may also contain a thumbnail version of an image, which is stored using a separate image representation.) If you pass YES in the *flag* parameter, and subsequently send a setSize: (page 1370) message to the receiver, all such image representations would be scaled to the same size. Scaling of bitmap images usually results in the interpolation of the bitmap data.

This method does not invalidate the caches of any of the receiver's image representations. The caches are not invalidated until you change the image size using a `setSize:` (page 1370) message. Scaling affects only the cached offscreen data for a given image representation.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
`- scalesWhenResized` (page 1362)

Related Sample Code
CocoaDragAndDrop
CompositeLab
FunkyOverlayWindow
MyCustomColorPicker
STUCAuthoringDeviceCocoaSample

Declared In
`NSImage.h`

setSize:

Sets the width and height of the image.

```
- (void)setSize:(NSSize)aSize
```

Parameters

aSize
The new size of the image, measured in points.

Discussion
The size of an `NSImage` object must be set before it can be used. If the size of the image hasn't already been set when an image representation is added, the size is taken from the image representation's data. For EPS images, the size is taken from the image's bounding box. For TIFF images, the size is taken from the `ImageLength` and `ImageWidth` attributes.

Changing the size of an `NSImage` after it has been used effectively resizes the image. Changing the size invalidates all its caches and frees them. When the image is next composited, the selected representation will draw itself in an offscreen window to recreate the cache.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- size` (page 1372)
`- initWithSize:` (page 1355)

Related Sample Code
ButtonMadness
DesktopImage
FunkyOverlayWindow

SourceView
STUCAuthoringDeviceCocoaSample

Declared In
NSImage.h

setTemplate:

Sets whether the image represents a template image.

```
- (void)setTemplate:(BOOL)isTemplate
```

Parameters

isTemplate

Specify YES if the image is a template image; otherwise, NO.

Discussion
Images you mark as template images should consist of only black and clear colors. You can use the alpha channel in the image to adjust the opacity of black content, however.

Template images are not intended to be used as standalone images. They are always mixed with other content and processed to create the desired appearance. You can mark an image as a "template image" to notify clients who care that the image contains only black and clear content. The most common use for template images is in image cells. For example, you might use a template image to provide the content for a button or segmented control. Cocoa cells take advantage of the nature of template images—that is, their simplified color scheme and use of transparency—to improve the appearance of the corresponding control in each of its supported states.

Availability
Available in Mac OS X v10.5 and later.

See Also
- isTemplate (page 1357)

Declared In
NSImage.h

setUsesEPSOnResolutionMismatch:

Sets whether EPS image representations are preferred when no other representations match the resolution of the device.

```
- (void)setUsesEPSOnResolutionMismatch:(BOOL)flag
```

Parameters

flag

YES if EPS image representations are preferred; otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- usesEPSOnResolutionMismatch (page 1374)

- setMatchesOnMultipleResolution: (page 1368)

Declared In
NSImage.h

size

Returns the size of the receiver.

- (NSSize)size

Return Value
The size of the receiver or (0.0, 0.0) if no size has been set and the size cannot be determined from any of the receiver's image representations.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setSize: (page 1370)

Related Sample Code
FunHouse
QTKitMovieShuffler
QuickLookSketch
RGB Image
Sketch-112

Declared In
NSImage.h

TIFFRepresentation

Returns a data object containing TIFF data for all of the image representations in the receiver.

- (NSData *)TIFFRepresentation

Return Value
A data object containing the TIFF data, or nil if the TIFF data could not be created.

Discussion
You can use the returned data object to write the TIFF data to a file. For each image representation, this method uses the TIFF compression option associated with that representation or NSTIFFCompressionNone, if no option is set.

If one of the receiver's image representations does not support the creation of TIFF data natively (PDF and EPS images, for example), this method creates the TIFF data from that representation's cached content.

Additional image formats can be saved by using the NSBitmapImageRep method representationUsingType:properties: (page 363).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `TIFFRepresentationUsingCompression:factor:` (page 1373)
`representationUsingType:properties:` (page 363) (NSBitmapImageRep)
`TIFFRepresentation` (page 366) (NSBitmapImageRep)
`TIFFRepresentationUsingCompression:factor:` (page 367) (NSBitmapImageRep)

Related Sample Code
GLSLShowpiece
ImageKitDemo
OpenCL_OceanWave
People
Sketch-112

Declared In
`NSImage.h`

TIFFRepresentationUsingCompression:factor:

Returns a data object containing TIFF data with the specified compression settings for all of the image representations in the receiver.

```
- (NSData *)TIFFRepresentationUsingCompression:(NSTIFFCompression)comp
    factor:(float)aFloat
```

Parameters

comp
The type of compression to use. For a list of values, see the constants in `NSBitmapImageRep`.

aFloat
Provides a hint for compression types that implement variable compression ratios. Currently, only JPEG compression uses a compression factor.

Return Value
A data object containing the TIFF data, or `nil` if the TIFF data could not be created.

Discussion
You can use the returned data object to write the TIFF data to a file. If the specified compression isn't applicable, no compression is used. If a problem is encountered during generation of the TIFF data, this method may raise an exception.

If one of the receiver's image representations does not support the creation of TIFF data natively (PDF and EPS images, for example), this method creates the TIFF data from that representation's cached content.

Additional image formats can be saved by using the `NSBitmapImageRep` method `representationUsingType:properties:` (page 363).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `TIFFRepresentation` (page 1372)
`representationUsingType:properties:` (page 363) (NSBitmapImageRep)
`TIFFRepresentation` (page 366) (NSBitmapImageRep)

`TIFFRepresentationUsingCompression:factor:` (page 367) (NSBitmapImageRep)

Related Sample Code
PDFKitLinker2

Declared In
`NSImage.h`

unlockFocus

Removes the focus from the receiver.

```
- (void)unlockFocus
```

Discussion
This message must be sent after a successful `lockFocus` or `lockFocusOnRepresentation:` message and the completion of any intermediate drawing commands. This method restores the focus to the previous owner, if any.

Do not send this message if the preceding call to lock focus raised an `NSImageCacheException`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
FunHouse
Image Difference
RGB Image
RGB ValueTransformers
Sketch-112

Declared In
`NSImage.h`

usesEPSOnResolutionMismatch

Returns a Boolean value indicating whether EPS representations are preferred when no other representations match the resolution of the device.

```
- (BOOL)usesEPSOnResolutionMismatch
```

Return Value
`YES` if EPS image representations are preferred; otherwise `NO`.

Discussion
The default value is `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setUsesEPSOnResolutionMismatch:` (page 1371)

- matchesOnMultipleResolution (page 1360)

Declared In
NSImage.h

Constants

Image Hint Dictionary Keys

These constants are a subset of the dictionary keys used in the hints dictionary for the methods CGImageForProposedRect:context:hints: (page 1339), bestRepresentationForRect:context:hints: (page 1338), drawInRect:fromRect:operation:fraction:respectFlipped:hints: (page 1347), and hitTestRect:withImageDestinationRect:context:hints:flipped: (page 1349). Additional hint keys are also valid including: Context Options in CIContext, and the entries in an NSScreen device description dictionary as described in deviceDescription (page 2317).

```
NSString *const NSImageHintCTM;
NSString *const NSImageHintInterpolation;
```

Constants

NSImageHintCTM
: Provides a context transform hint. The value for this key is an NSAffineTransform.

  Available in Mac OS X v10.6 and later.

  Declared in NSImage.h.

NSImageHintInterpolation
: Provides an interpolation hint. The value for this key is an NSNumber with an NSImageInterpolation (page 1313) value.

  Available in Mac OS X v10.6 and later.

  Declared in NSImage.h.

NSCompositingOperation

These constants specify compositing operators described in terms of having source and destination images, each having an opaque and transparent region. The destination image after the operation is defined in terms of the source and destination before images.

```
enum {
    NSCompositeClear           = 0,
    NSCompositeCopy            = 1,
    NSCompositeSourceOver      = 2,
    NSCompositeSourceIn        = 3,
    NSCompositeSourceOut       = 4,
    NSCompositeSourceAtop      = 5,
    NSCompositeDestinationOver = 6,
    NSCompositeDestinationIn   = 7,
    NSCompositeDestinationOut  = 8,
    NSCompositeDestinationAtop = 9,
    NSCompositeXOR             = 10,
    NSCompositePlusDarker      = 11,
    NSCompositeHighlight       = 12,
    NSCompositePlusLighter     = 13
}
typedef NSUInteger NSCompositingOperation;
```

Constants

`NSCompositeClear`

Transparent. (`R = 0`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeCopy`

Source image. (`R = S`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeSourceOver`

Source image wherever source image is opaque, and destination image elsewhere. (`R = S + D*(1 - Sa)`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeSourceIn`

Source image wherever both images are opaque, and transparent elsewhere. (`R = S*Da`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeSourceOut`

Source image wherever source image is opaque but destination image is transparent, and transparent elsewhere. (`R = S*(1 - Da)`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeSourceAtop`

Source image wherever both images are opaque, destination image wherever destination image is opaque but source image is transparent, and transparent elsewhere. (`R = S*Da + D*(1 - Sa)`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeDestinationOver`
Destination image wherever destination image is opaque, and source image elsewhere. (`R = S*(1 - Da) + D`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeDestinationIn`
Destination image wherever both images are opaque, and transparent elsewhere. (`R = D*Sa`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeDestinationOut`
Destination image wherever destination image is opaque but source image is transparent, and transparent elsewhere. (`R = D*(1 - Sa)`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeDestinationAtop`
Destination image wherever both images are opaque, source image wherever source image is opaque but destination image is transparent, and transparent elsewhere. (`R = S*(1 - Da) + D*Sa`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeXOR`
Exclusive OR of source and destination images. (`R = S*(1 - Da) + D*(1 - Sa)`)

Works only with black and white images and is not recommended for color contexts.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositePlusDarker`
Sum of source and destination images, with color values approaching 0 as a limit. (`R = MAX(0, (1 - D) + (1 - S))`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositeHighlight`

Source image wherever source image is opaque, and destination image elsewhere. (Deprecated. Mapped to `NSCompositeSourceOver`.)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCompositePlusLighter`
Sum of source and destination images, with color values approaching 1 as a limit. (`R = MIN(1, S + D)`)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

Discussion
These compositing operators are defined in and used by `compositeToPoint:fromRect:operation:` (page 1340), `compositeToPoint:operation:` (page 1342), `compositeToPoint:fromRect:operation:fraction:` (page 1341), `compositeToPoint:operation:fraction:` (page 1343),

`drawAtPoint:fromRect:operation:fraction:` (page 1346), and `drawInRect:fromRect:operation:fraction:` (page 1347). They are also used by drawing methods in other classes that take a compositing operator.

The equations after each constant represent the mathematical formulas used to calculate the color value of the resulting pixel. Table 61-2 lists the meaning of each placeholder value in the equations.

Table 61-2 Placeholder values for compositing equations

Para	Para
`R`	The premultiplied result color.
`S`	The source color
`D`	The destination color
`Sa`	The alpha value of the source color
`Da`	The alpha value of the destination color

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphics.h`

NSImageLoadStatus

These constants are status values passed to the incremental loading delegate method `image:didLoadRepresentation:withStatus:` (page 3694).

```
enum {
    NSImageLoadStatusCompleted,
    NSImageLoadStatusCancelled,
    NSImageLoadStatusInvalidData,
    NSImageLoadStatusUnexpectedEOF,
    NSImageLoadStatusReadError
}
typedef NSUInteger NSImageLoadStatus;
```

Constants

`NSImageLoadStatusCompleted`

Enough data has been provided to completely decompress the image.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageLoadStatusCancelled`

Image loading was canceled.

The image contains the portions of the data that have already been successfully decompressed, if any.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageLoadStatusInvalidData`
: An error occurred during image decompression.

  The image data is probably corrupt. The image contains the portions of the data that have already been successfully decompressed, if any.

  Available in Mac OS X v10.2 and later.

  Declared in `NSImage.h`.

`NSImageLoadStatusUnexpectedEOF`
: Not enough data was available for full decompression of the image.

  The image contains the portions of the data that have already been successfully decompressed, if any.

  Available in Mac OS X v10.2 and later.

  Declared in `NSImage.h`.

`NSImageLoadStatusReadError`
: Not enough data was available for full decompression of the image.

  The image contains the portions of the data that have already been successfully decompressed, if any.

  Available in Mac OS X v10.2 and later.

  Declared in `NSImage.h`.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSImage.h`

NSImageCacheMode

These constants specify the caching policy on a per `NSImage` basis. The caching policy is set using `cacheMode` (page 1339) and `setCacheMode:` (page 1365).

```
enum {
    NSImageCacheDefault,
    NSImageCacheAlways,
    NSImageCacheBySize,
    NSImageCacheNever
}
typedef NSUInteger NSImageCacheMode;
```

Constants
`NSImageCacheDefault`
: Caching is unspecified.

  Use the image rep's default.

  Available in Mac OS X v10.2 and later.

  Declared in `NSImage.h`.

`NSImageCacheAlways`
: Always generate a cache when drawing.

  Available in Mac OS X v10.2 and later.

  Declared in `NSImage.h`.

`NSImageCacheBySize`
Cache if cache size is smaller than the original data.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

`NSImageCacheNever`
Never cache; always draw direct.

Available in Mac OS X v10.2 and later.

Declared in `NSImage.h`.

Discussion
The following table specifies the default caching policy for the various types of image representation.

Image Rep Class	Default caching policy
`NSBitmapImageRep`	`NSImageCacheBySize`. Cache if bitmap is 32-bits in 16-bit world or greater than 72 dpi.
`NSPICTImageRep`	`NSImageCacheBySize`. Same reasoning as `NSBitmapImageRep` in the event the PICT contains a bitmap.
`NSPDFImageRep`	`NSImageCacheAlways`
`NSCIImageRep`	`NSImageCacheBySize`. Cache if the bitmap depth does not match the screen depth or the resolution is greater than 72 dpi.
`NSEPSImageRep`	`NSImageCacheAlways`
`NSCustomImageRep`	`NSImageCacheAlways`

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSImage.h`

Image Template Constants

Images representing standard artwork and icons that you can use in your applications

```
NSString *const NSImageNameQuickLookTemplate;
NSString *const NSImageNameBluetoothTemplate;
NSString *const NSImageNameIChatTheaterTemplate;
NSString *const NSImageNameSlideshowTemplate;
NSString *const NSImageNameActionTemplate;
NSString *const NSImageNameSmartBadgeTemplate;
NSString *const NSImageNamePathTemplate;
NSString *const NSImageNameInvalidDataFreestandingTemplate;
NSString *const NSImageNameLockLockedTemplate;
NSString *const NSImageNameLockUnlockedTemplate;
NSString *const NSImageNameGoRightTemplate;
NSString *const NSImageNameGoLeftTemplate;
NSString *const NSImageNameRightFacingTriangleTemplate;
NSString *const NSImageNameLeftFacingTriangleTemplate;
NSString *const NSImageNameAddTemplate;
NSString *const NSImageNameRemoveTemplate;
NSString *const NSImageNameRevealFreestandingTemplate;
NSString *const NSImageNameFollowLinkFreestandingTemplate;
NSString *const NSImageNameEnterFullScreenTemplate;
NSString *const NSImageNameExitFullScreenTemplate;
NSString *const NSImageNameStopProgressTemplate;
NSString *const NSImageNameStopProgressFreestandingTemplate;
NSString *const NSImageNameRefreshTemplate;
NSString *const NSImageNameRefreshFreestandingTemplate;
NSString *const NSImageNameFolder;
NSString *const NSImageNameTrashEmpty;
NSString *const NSImageNameTrashFull;
NSString *const NSImageNameHomeTemplate;
NSString *const NSImageNameBookmarksTemplate;
NSString *const NSImageNameCaution;
NSString *const NSImageNameStatusAvailable;
NSString *const NSImageNameStatusPartiallyAvailable;
NSString *const NSImageNameStatusUnavailable;
NSString *const NSImageNameStatusNone;
NSString *const NSImageNameApplicationIcon;
NSString *const NSImageNameMenuOnStateTemplate;
NSString *const NSImageNameMenuMixedStateTemplate;
NSString *const NSImageNameUserGuest;
NSString *const NSImageNameMobileMe;
```

Constants

`NSImageNameQuickLookTemplate`

A Quick Look template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameBluetoothTemplate`

A Bluetooth template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameIChatTheaterTemplate`

An iChat Theater template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameSlideshowTemplate`

A slideshow template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameActionTemplate`

An action menu template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameSmartBadgeTemplate`

A badge for a “smart” item. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNamePathTemplate`

A path button template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameInvalidDataFreestandingTemplate`

An invalid data template image. Place this icon to the right of any fields containing invalid data. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameLockLockedTemplate`

A locked lock template image. Use to indicate locked content. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameLockUnlockedTemplate`

An unlocked lock template image. Use to indicate modifiable content that can be locked. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameGoRightTemplate`

A “go forward” template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameGoLeftTemplate`

A “go back” template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameRightFacingTriangleTemplate`

A generic right-facing triangle template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameLeftFacingTriangleTemplate`

A generic left-facing triangle template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameAddTemplate`

An add item template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameRemoveTemplate`

A remove item template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameRevealFreestandingTemplate`

A reveal contents template image. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameFollowLinkFreestandingTemplate`

A link template image. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameEnterFullScreenTemplate`

An enter full-screen mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameExitFullScreenTemplate`

An exit full-screen mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameStopProgressTemplate`

A stop progress button template image.

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameStopProgressFreestandingTemplate`

A stop progress template image. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameRefreshTemplate`

A refresh template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameRefreshFreestandingTemplate`

A refresh template image. You can use this image to implement a borderless button. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

Discussion
To access these images, pass the specified constant to the `imageNamed:` (page 1330) method.

Images with the word "Template" in their title identify shapes that are not intended as standalone images. You would typically use these icons as the custom image for a button, or you might apply them to a cell in a control. For example, you might use the `NSImageNameLockLockedTemplate` image to indicate an item is not modifiable. Template images should use black and clear colors only and it is fine to include varying levels of alpha.

Images with the word "Freestanding" in their title can be used to implement borderless buttons. You do not need to include any extra bezel artwork behind such images.

You should always use named images according to their intended purpose, and not according to how the image appears when loaded. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the "ImageName" portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameRefreshFreestandingTemplate` would correspond to an image named "NSRefreshFreestandingTemplate" in Interface Builder.

Declared In
`NSImage.h`

Multiple Documents Drag Image

Drag images you can use in your applications. To access this image, pass the specified constant to the `imageNamed:` (page 1330) method.

```
NSString *const NSImageNameMultipleDocuments;
```

Constants

`NSImageNameMultipleDocuments`

A drag image for multiple items. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

Discussion

You can use this icon as the drag image when dragging multiple items. You should not use this image for any other intended purpose, however. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the "ImageName" portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameMultipleDocuments` would correspond to an image named "NSMultipleDocuments" in Interface Builder.

Sharing Permissions Named Images

Images representing sharing permission icons that you can use in your applications. To access this image, pass the specified constant to the `imageNamed:` (page 1330) method.

```
NSString *const NSImageNameUser;
NSString *const NSImageNameUserGroup;
NSString *const NSImageNameEveryone;
```

Constants

`NSImageNameUser`

Permissions for a single user. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameUserGroup`

Permissions for a group of users. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameEveryone`

Permissions for all users. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

Discussion
You should use these images to reflect user and group permission or sharing information. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the "ImageName" portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameEveryone` would correspond to an image named "NSEveryone" in Interface Builder.

System Entity Images

Images representing Finder items. To access this image, pass the specified constant to the `imageNamed:` (page 1330) method.

```
NSString *const NSImageNameBonjour;
NSString *const NSImageNameDotMac;
NSString *const NSImageNameComputer;
NSString *const NSImageNameFolderBurnable;
NSString *const NSImageNameFolderSmart;
NSString *const NSImageNameNetwork;
```

Constants

`NSImageNameBonjour`

A Bonjour icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameDotMac`

A Dot Mac icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameComputer`

A computer icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameFolderBurnable`

A burnable folder icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameFolderSmart`

A smart folder icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameNetwork`

A network icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

Discussion

You should use these images to reflect specific elements of the Mac OS X environment. For example, you might use the burnable folder icon if your software allows the user to organize content for burning onto an optical disk. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameNetwork` would correspond to an image named “NSNetwork” in Interface Builder.

Declared In

`NSImage.h`

Toolbar Named Images

Images that you can use in application toolbars. To access this image, pass the specified constant to the `imageNamed:` (page 1330) method.

```
NSString *const NSImageNameUserAccounts;
NSString *const NSImageNamePreferencesGeneral;
NSString *const NSImageNameAdvanced;
NSString *const NSImageNameInfo;
NSString *const NSImageNameFontPanel;
NSString *const NSImageNameColorPanel;
```

Constants

`NSImageNameUserAccounts`

User account toolbar icon. Use in a preferences window only. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNamePreferencesGeneral`

General preferences toolbar icon. Use in a preferences window only. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameAdvanced`

Advanced preferences toolbar icon. Use in a preferences window only. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameInfo`

An information toolbar icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameFontPanel`

A font panel toolbar icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameColorPanel`

A color panel toolbar icon. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameFolder`

A folder image. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameTrashEmpty`

An image of the empty trash can. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameTrashFull`

An image of the full trash can. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

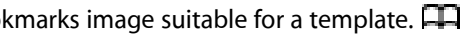
`NSImageNameHomeTemplate`

Home image suitable for a template. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameBookmarksTemplate`

Bookmarks image suitable for a template. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameCaution`

Caution Image. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameStatusAvailable`

Small green indicator, similar to iChat's available image. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameStatusPartiallyAvailable`

Small yellow indicator, similar to iChat's idle image. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameStatusUnavailable`

Small red indicator, similar to iChat's unavailable image. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameStatusNone`

Small clear indicator. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameApplicationIcon`

Generic application icon. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameMenuOnStateTemplate`

A check mark. Drawing these outside of menus is discouraged. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameMenuMixedStateTemplate`

A horizontal dash. Drawing these outside of menus is discouraged.

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameUserGuest`

Shaded user figure. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

`NSImageNameMobileMe`

MobileMe logo. Note that this is preferred to using the `NSImageNameDotMac` (page 1386) image, although that image is not expected to be deprecated. 

Available in Mac OS X v10.6 and later.

Declared in `NSImage.h`.

Discussion
You should use these images as icons for toolbar items. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

Constants that end in the word "Template" name black and clear images that return `YES` for `isTemplate` (page 1357). These images can be processed into variants appropriate for different situations. For example, these images can invert in a selected table view row. See `setTemplate:` (page 1371): for more comments. These images are inappropriate for display without further processing, but `NSCell` and its subclasses will perform the processing.

Some images also contain the word "Freestanding". This indicates that an image is appropriate for use as a borderless button, it doesn't need any extra bezel artwork behind it. For example, Safari uses `NSImageNameStopProgressFreestandingTemplate` (page 1383) as the stop button in a button on its toolbar, while it uses `NSImageNameStopProgressFreestandingTemplate` (page 1383) in the downloads window where it appears inline with a progress indicator.

The string value for each constant is equal to the constant name without the “ImageName” portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameColorPanel` would correspond to an image named “NSColorPanel” in Interface Builder.

Declared In
`NSImage.h`

View Type Template Images

Images used in segmented controls to switch the current view type. To access this image, pass the specified constant to the `imageNamed:` (page 1330) method.

```
NSString *const NSImageNameIconViewTemplate;
NSString *const NSImageNameListViewTemplate;
NSString *const NSImageNameColumnViewTemplate;
NSString *const NSImageNameFlowViewTemplate;
```

Constants

`NSImageNameIconViewTemplate`

An icon view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameListViewTemplate`

A list view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameColumnViewTemplate`

A column view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

`NSImageNameFlowViewTemplate`

A cover flow view mode template image. 

Available in Mac OS X v10.5 and later.

Declared in `NSImage.h`.

Discussion

Images with the word "Template" in their title identify shapes that are not intended as standalone images. You would typically use these icons as the custom image for a button, or you might apply them to a cell in a control. For example, you might use the `NSImageNameIconViewTemplate` image to indicate an item is not modifiable. Template images should use black and clear colors only and it is fine to include varying levels of alpha.

You should use these images in conjunction with the buttons (usually part of a segmented control) that change the current viewing mode. The appearance of images can change between releases. If you use an image for its intended purpose (and not because of how it looks), your code should look correct from release to release.

The size and aspect ratio of system images may change from release to release. In some situations, you should explicitly resize images as appropriate for your use. If you use these images in conjunction with an `NSButtonCell` object, however, you can use the `setImageScaling:` method of the cell to control scaling instead. Similarly, for an `NSSegmentedCell` object, you can use the `setImageScaling:forSegment:` method to control scaling.

The string value for each constant is equal to the constant name without the "ImageName" portion. You might need this information to locate images by name in Interface Builder. For example, the constant `NSImageNameFlowViewTemplate` would correspond to an image named "NSFlowViewTemplate" in Interface Builder.

Declared In

`NSImage.h`

NSImageCell Class Reference

Inherits from	NSCell : NSObject
Conforms to	NSCoding NSCopying NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSImageCell.h
Companion guides	Image Views Matrix Programming Guide Table View Programming Guide
Related sample code	AnimatedTableView Cropped Image STUCAuthoringDeviceCocoaSample

Overview

An `NSImageCell` object displays a single image (encapsulated in an `NSImage` object) in a frame. This class provides methods for choosing the frame and for aligning and scaling the image to fit the frame.

The object value of an `NSImageCell` object must be an `NSImage` object, so if you use the `setObjectValue:` (page 592) method of `NSCell`, be sure to supply an `NSImage` object as an argument. Because an `NSImage` object does not need to be converted for display, do not use the `NSCell` methods relating to formatters.

An `NSImageCell` object is usually associated with some kind of control object. For example, an `NSMatrix` or an `NSTableView`.

Designated Initializers

When subclassing `NSImageCell` you must implement all of the designated initializers. Those methods are: init, `initWithCoder:`, `initTextCell:` (page 564), and `initImageCell:` (page 563).

Tasks

Aligning and Scaling the Image

- `imageAlignment` (page 1394)
  Returns the alignment of the receiver's image relative to its frame.
- `setImageAlignment:` (page 1395)
  Sets the alignment of the image in its frame.
- `imageScaling` (page 1395)
  Returns the scaling mode used to fit the receiver's image into the frame.
- `setImageScaling:` (page 1396)
  Sets the scaling mode used to fit the receiver's image into the frame.

Choosing the Frame

- `imageFrameStyle` (page 1394)
  Returns the style of the frame that borders the image.
- `setImageFrameStyle:` (page 1396)
  Sets the style of the frame that borders the image.

Instance Methods

imageAlignment

Returns the alignment of the receiver's image relative to its frame.

```
- (NSImageAlignment)imageAlignment
```

Return Value
One of the image alignment constants. For a list of possible values, see `NSImageAlignment` (page 1396). The default value is `NSImageAlignCenter`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setImageAlignment:` (page 1395)

Declared In
`NSImageCell.h`

imageFrameStyle

Returns the style of the frame that borders the image.

```
- (NSImageFrameStyle)imageFrameStyle
```

Return Value
One of the frame style constants. For a list of frame styles, see `NSImageFrameStyle` (page 1398). The default value is `NSImageFrameNone`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setImageFrameStyle:` (page 1396)

Declared In
`NSImageCell.h`

imageScaling

Returns the scaling mode used to fit the receiver's image into the frame.

```
- (NSImageScaling)imageScaling
```

Return Value
One of the image scaling constants. For a list of possible values, see `NSImageScaling` (page 617). The default value is `NSImageScaleProportionallyDown` (page 617).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setImageScaling:` (page 1396)

Declared In
`NSImageCell.h`

setImageAlignment:

Sets the alignment of the image in its frame.

```
- (void)setImageAlignment:(NSImageAlignment)alignment
```

Parameters
alignment
One of the image alignment constants. For a list of possible values, see `NSImageAlignment` (page 1396).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- imageAlignment` (page 1394)

Declared In
`NSImageCell.h`

setImageFrameStyle:

Sets the style of the frame that borders the image.

```
- (void)setImageFrameStyle:(NSImageFrameStyle)frameStyle
```

Parameters

frameStyle

One of the frame style constants. For a list of frame styles, see `NSImageFrameStyle` (page 1398).

Availability

Available in Mac OS X v10.0 and later.

See Also

`- imageFrameStyle` (page 1394)

Declared In

`NSImageCell.h`

setImageScaling:

Sets the scaling mode used to fit the receiver's image into the frame.

```
- (void)setImageScaling:(NSImageScaling)scaling
```

Parameters

scaling

One of the image scaling constants. For a list of possible values, see `NSImageScaling` (page 617).

Availability

Available in Mac OS X v10.0 and later.

See Also

`- imageScaling` (page 1395)

Declared In

`NSImageCell.h`

Constants

NSImageAlignment

These constants allow you to specify the location of the image in the frame and are used by `imageAlignment` (page 1394) and `setImageAlignment:` (page 1395).

```
enum {
    NSImageAlignCenter = 0,
    NSImageAlignTop,
    NSImageAlignTopLeft,
    NSImageAlignTopRight,
    NSImageAlignLeft,
    NSImageAlignBottom,
    NSImageAlignBottomLeft,
    NSImageAlignBottomRight,
    NSImageAlignRight
};
typedef NSUInteger NSImageAlignment;
```

Constants

`NSImageAlignCenter`
Center the image in the cell.
Available in Mac OS X v10.0 and later.
Declared in `NSImageCell.h`.

`NSImageAlignTop`
Position the image along the top edge of the cell.
Available in Mac OS X v10.0 and later.
Declared in `NSImageCell.h`.

`NSImageAlignTopLeft`
Align the image with the top and left edges of the cell.
Available in Mac OS X v10.0 and later.
Declared in `NSImageCell.h`.

`NSImageAlignTopRight`
Align the image with the top and right edges of the cell.
Available in Mac OS X v10.0 and later.
Declared in `NSImageCell.h`.

`NSImageAlignLeft`
Align the image with the left edge of the cell.
Available in Mac OS X v10.0 and later.
Declared in `NSImageCell.h`.

`NSImageAlignBottom`
Align the image with the bottom edge of the cell.
Available in Mac OS X v10.0 and later.
Declared in `NSImageCell.h`.

`NSImageAlignBottomLeft`
Align the image with the bottom and left edges of the cell.
Available in Mac OS X v10.0 and later.
Declared in `NSImageCell.h`.

`NSImageAlignBottomRight`
Align the image with the bottom and right edges of the cell.
Available in Mac OS X v10.0 and later.
Declared in `NSImageCell.h`.

`NSImageAlignRight`
Position the image along the right edge of the cell.

Available in Mac OS X v10.0 and later.

Declared in `NSImageCell.h`.

Declared In
`NSImageCell.h`

NSImageFrameStyle

These constants allow you to specify the kind of frame bordering the image and are used by `imageFrameStyle` (page 1394) and `setImageFrameStyle:` (page 1396). (Deprecated. These constants are obsolete, and are not compliant with the *Apple Human Interface Guidelines*.)

```
enum {
    NSImageFrameNone = 0,
    NSImageFramePhoto,
    NSImageFrameGrayBezel,
    NSImageFrameGroove,
    NSImageFrameButton
};
typedef NSUInteger NSImageFrameStyle;
```

Constants
`NSImageFrameNone`
An invisible frame

Available in Mac OS X v10.0 and later.

Declared in `NSImageCell.h`.

`NSImageFramePhoto`
A thin black outline and a dropped shadow

Available in Mac OS X v10.0 and later.

Declared in `NSImageCell.h`.

`NSImageFrameGrayBezel`
A gray, concave bezel that makes the image look sunken

Available in Mac OS X v10.0 and later.

Declared in `NSImageCell.h`.

`NSImageFrameGroove`
A thin groove that looks etched around the image

Available in Mac OS X v10.0 and later.

Declared in `NSImageCell.h`.

`NSImageFrameButton`
A convex bezel that makes the image stand out in relief, like a button

Available in Mac OS X v10.0 and later.

Declared in `NSImageCell.h`.

Declared In
`NSImageCell.h`

NSImageRep Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSImageRep.h
Companion guide	Cocoa Drawing Guide
Related sample code	LayerBackedOpenGLView NSOpenGL Fullscreen Reducer

Overview

The `NSImageRep` class is a semiabstract superclass ("semi" because it has some instance variables and implementation of its own). Each of its subclasses knows how to draw an image from a particular kind of source data. While an `NSImageRep` subclass can be used directly, it is typically through an `NSImage` object. An `NSImage` object manages a group of image representations, choosing the best one for the current output device.

Tasks

Creating an NSImageRep

`+ imageRepsWithContentsOfFile:` (page 1405)
Creates and returns an array of image representation objects initialized using the contents of the specified file.

`+ imageRepsWithPasteboard:` (page 1407)
Creates and returns an array of image representation objects initialized using the contents of the pasteboard.

+ `imageRepsWithContentsOfURL:` (page 1406)
Creates and returns an array of image representation objects initialized using the contents of the specified URL.

+ `imageRepWithContentsOfFile:` (page 1408)
Creates and returns an image representation object using the contents of the specified file.

+ `imageRepWithPasteboard:` (page 1409)
Creates and returns an image representation object using the contents of the specified pasteboard.

+ `imageRepWithContentsOfURL:` (page 1409)
Creates and returns an image representation object using the data at the specified URL

Determining the Supported Image Types

+ `canInitWithData:` (page 1402)
Returns a Boolean value indicating whether the receiver can initialize itself from the specified data.

+ `canInitWithPasteboard:` (page 1403)
Returns a Boolean value indicating whether the receiver can initialize itself from the data on the specified pasteboard.

+ `imageTypes` (page 1410)
Returns an array of UTI strings identifying the image types supported by the receiver, either directly or through a user-installed filter service.

+ `imageUnfilteredTypes` (page 1412)
Returns an array of UTI strings identifying the image types supported directly by the receiver.

+ `imageFileTypes` (page 1403)
Returns the file types supported by `NSImageRep` or one of its subclasses.

+ `imagePasteboardTypes` (page 1404)
Returns the pasteboard types supported by `NSImageRep` or one of its subclasses.

+ `imageUnfilteredFileTypes` (page 1410)
Returns the list of file types supported directly by the receiver.

+ `imageUnfilteredPasteboardTypes` (page 1411)
Returns the list of pasteboard types supported directly by the receiver.

Setting the Size of the Image

- `setSize:` (page 1423)
Sets the size of the image representation to the specified value.

- `size` (page 1424)
Returns the size of the image representation.

Specifying Information About the Representation

- `bitsPerSample` (page 1414)
Returns the number of bits per sample in the receiver.

- `colorSpaceName` (page 1415)
  Returns the name of the receiver's color space.
- `hasAlpha` (page 1418)
  Returns a Boolean value indicating whether the receiver has an alpha channel.
- `isOpaque` (page 1419)
  Returns a Boolean value indicating whether the receiver is opaque.
- `pixelsHigh` (page 1419)
  Returns the height of the image, measured in pixels.
- `pixelsWide` (page 1420)
  Returns the width of the image, measured in pixels.
- `setAlpha:` (page 1420)
  Informs the receiver that its image data has an alpha component.
- `setBitsPerSample:` (page 1421)
  Informs the receiver that its image data has the specified number of bits for each component of a pixel.
- `setColorSpaceName:` (page 1421)
  Informs the receiver of the color space used by the image data.
- `setOpaque:` (page 1422)
  Sets whether the receiver's image is opaque.
- `setPixelsHigh:` (page 1422)
  Informs the receiver of the image data height.
- `setPixelsWide:` (page 1423)
  Informs the receiver of the image data width.

Getting a CGImage

- `CGImageForProposedRect:context:hints:` (page 1414)
  Returns a `CGImage` capturing the drawing of the receiver.

Drawing the Image

- `draw` (page 1415)
  Implemented by subclasses to draw the image in the current coordinate system.
- `drawAtPoint:` (page 1416)
  Draws the receiver's image data at the specified point in the current coordinate system.
- `drawInRect:` (page 1417)
  Draws the image, scaling it (as needed) to fit the specified rectangle.
- `drawInRect:fromRect:operation:fraction:respectFlipped:hints:` (page 1417)
  Draws all or part of the image in the specified rectangle in the current coordinate system.

Managing NSImageRep Subclasses

`+ imageRepClassForType:` (page 1405)
Returns the `NSImageRep` subclass that handles image data for the specified UTI.

`+ imageRepClassForData:` (page 1404)
Returns the `NSImageRep` subclass that handles the specified type of data.

`+ imageRepClassForFileType:` (page 1404)
Returns the `NSImageRep` subclass that handles data with the specified type.

`+ imageRepClassForPasteboardType:` (page 1405)
Returns the `NSImageRep` subclass that handles data with the specified pasteboard type.

`+ registeredImageRepClasses` (page 1412)
Returns an array containing the registered `NSImageRep` classes.

`+ registerImageRepClass:` (page 1413)
Adds the specified class to the registry of available `NSImageRep` subclasses.

`+ unregisterImageRepClass:` (page 1413)
Removes the specified `NSImageRep` subclass from the registry of available image representations.

Class Methods

canInitWithData:

Returns a Boolean value indicating whether the receiver can initialize itself from the specified data.

```
+ (BOOL)canInitWithData:(NSData *)data
```

Parameters

data
The image data.

Return Value
`YES` if the receiver understands the format of the specified data and can use it to initialize itself; otherwise, `NO`.

Discussion
This method should be overridden by subclasses. Note that this method does not need to do a comprehensive check of the image data; it should return `NO` only if it knows it cannot initialize itself from the data.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ canInitWithPasteboard:` (page 1403)

Declared In
`NSImageRep.h`

canInitWithPasteboard:

Returns a Boolean value indicating whether the receiver can initialize itself from the data on the specified pasteboard.

```
+ (BOOL)canInitWithPasteboard:(NSPasteboard *)pasteboard
```

Parameters

pasteboard

The pasteboard containing the image data.

Return Value

`YES` if the receiver understands the format of the specified data and can use it to initialize itself; otherwise, `NO`.

Discussion

This method invokes the `imageUnfilteredPasteboardTypes` (page 1411) class method and checks the list of types returned by that method against the data types in *pasteboard*. If it finds a match, it returns `YES`. When creating a subclass of `NSImageRep` that accepts image data from a non-default pasteboard type, override the `imageUnfilteredPasteboardTypes` (page 1411) method to assure this method returns the correct response.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ canInitWithData:` (page 1402)

Declared In

`NSImageRep.h`

imageFileTypes

Returns the file types supported by `NSImageRep` or one of its subclasses.

```
+ (NSArray *)imageFileTypes
```

Return Value

An array of `NSString` objects, each of which contains a filename extension or HFS file type of a supported format.

Discussion

The list includes both those types returned by the `imageUnfilteredFileTypes` (page 1410) class method plus those that can be converted to a supported type by a user-installed filter service. The returned file types can include encoded HFS file types as well as filename extensions.

Don't override this method when subclassing `NSImageRep`—it always returns a valid list for any subclass of `NSImageRep` that correctly overrides the `imageUnfilteredFileTypes` (page 1410) method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSImageRep.h`

imagePasteboardTypes

Returns the pasteboard types supported by `NSImageRep` or one of its subclasses.

```
+ (NSArray *)imagePasteboardTypes
```

Return Value
An array of `NSString` objects, each of which contains a supported pasteboard format.

Discussion
The list includes both those types returned by the `imageUnfilteredPasteboardTypes` (page 1411) class method plus those that can be converted to a supported type by a user-installed filter service. Don't override this method when subclassing `NSImageRep`—it always returns a valid list for any subclass of `NSImageRep` that correctly overrides the `imageUnfilteredPasteboardTypes` (page 1411) method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSImageRep.h`

imageRepClassForData:

Returns the `NSImageRep` subclass that handles the specified type of data.

```
+ (Class)imageRepClassForData:(NSData *)data
```

Parameters

data
The image data.

Return Value
A `Class` object for the image representation that can handle the data, or `nil` if no image representation could handle the data.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSImageRep.h`

imageRepClassForFileType:

Returns the `NSImageRep` subclass that handles data with the specified type.

```
+ (Class)imageRepClassForFileType:(NSString *)type
```

Parameters

type
A string containing the filename extension or an encoded HFS type.

Return Value
A `Class` object for the image representation that can handle the type of data, or `nil` if no image representation could handle the type.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSImageRep.h`

imageRepClassForPasteboardType:

Returns the `NSImageRep` subclass that handles data with the specified pasteboard type.

```
+ (Class)imageRepClassForPasteboardType:(NSString *)type
```

Parameters

type

The pasteboard type.

Return Value
A `Class` object for the image representation that can handle the specified pasteboard type, or `nil` if no image representation could handle the type.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSImageRep.h`

imageRepClassForType:

Returns the `NSImageRep` subclass that handles image data for the specified UTI.

```
+ (Class)imageRepClassForType:(NSString *)type
```

Parameters

type

The UTI string identifying the desired image type. Some sample image-related UTI strings include "`public.image`", "`public.jpeg`", and "`public.tiff`". For a list of supported types, see `UTCoreTypes.h`.

Return Value
A `Class` object for the image representation that can handle the UTI, or `nil` if no image representation could handle the data.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSImageRep.h`

imageRepsWithContentsOfFile:

Creates and returns an array of image representation objects initialized using the contents of the specified file.

```
+ (NSArray *)imageRepsWithContentsOfFile:(NSString *)filename
```

Parameters

filename

A full or relative pathname specifying the file to open. This string should include the filename extension.

Return Value

An array of image representation objects. The array contains one object for each image in the specified file.

Discussion

If sent to the `NSImageRep` class object, this method returns an array of objects (all newly allocated instances of a subclass of `NSImageRep`, chosen through the use of `imageRepClassForFileType:` (page 1404)) that have been initialized with the contents of the file. If sent to a subclass of `NSImageRep` that recognizes the file type, this method returns an array of objects (all instances of that subclass) that have been initialized with the contents of the file.

This method returns `nil` in any of the following cases:

- The message is sent to the `NSImageRep` class object and there are no subclasses in the `NSImageRep` class registry that handle the data in the file.
- The message is sent to a subclass of `NSImageRep` and that subclass cannot handle the data in the file.
- The `NSImageRep` subclass is unable to initialize itself with the contents of *filename*.

The `NSImageRep` subclass is initialized by creating an `NSData` object based on the contents of the file and passing it to the `imageRepsWithData:` method of the subclass. By default, the files handled include those with the extensions "`tiff`", "`gif`", "`jpg`", "`pict`", "`pdf`", and "`eps`".

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ imageFileTypes` (page 1403)

Declared In

`NSImageRep.h`

imageRepsWithContentsOfURL:

Creates and returns an array of image representation objects initialized using the contents of the specified URL.

```
+ (NSArray *)imageRepsWithContentsOfURL:(NSURL *)aURL
```

Parameters

aURL

The URL pointing to the image data.

Return Value

An array of image representation objects. The array contains one object for each image in the data at the specified URL.

Discussion
If sent to the `NSImageRep` class object, this method returns an array of objects (all newly allocated instances of a subclass of `NSImageRep`) that have been initialized with the contents of the specified URL. If sent to a subclass of `NSImageRep` that recognizes the data at the specified URL, it returns an array of objects (all instances of that subclass) that have been initialized with the contents of that URL.

This method returns `nil` in any of the following cases:

- The message is sent to the `NSImageRep` class object and there are no subclasses in the `NSImageRep` class registry that handle data in the specified URL.
- The message is sent to a subclass of `NSImageRep` and that subclass cannot handle data in the specified URL.
- The `NSImageRep` subclass is unable to initialize itself with the contents of the specified URL.

The `NSImageRep` subclass is initialized by creating an `NSData` object based on the contents of the specified URL and passing it to the `imageRepsWithData:` method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSImageRep.h`

imageRepsWithPasteboard:

Creates and returns an array of image representation objects initialized using the contents of the pasteboard.

```
+ (NSArray *)imageRepsWithPasteboard:(NSPasteboard *)pasteboard
```

Parameters
pasteboard
The pasteboard containing the image data.

Return Value
An array of image representation objects. The array contains one object for each image in the specified pasteboard.

Discussion
If sent to the `NSImageRep` class object, this method returns an array of objects (all newly-allocated instances of a subclass of `NSImageRep`) that have been initialized with the data in the specified pasteboard. If sent to a subclass of `NSImageRep` that recognizes the pasteboard data, it returns an array of objects (all instances of that subclass) initialized with the pasteboard data.

This method returns `nil` in any of the following cases:

- The message is sent to the `NSImageRep` class object and there are no subclasses in the `NSImageRep` class registry that handle the pasteboard data.
- The message is sent to a subclass of `NSImageRep` and that subclass cannot handle the pasteboard data.
- The `NSImageRep` subclass is unable to initialize itself with the contents the pasteboard.

The NSImageRep subclass is initialized by creating an NSData object based on the data in *pasteboard* and passing it to the imageRepsWithData: method.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ imagePasteboardTypes (page 1404)

Declared In
NSImageRep.h

imageRepWithContentsOfFile:

Creates and returns an image representation object using the contents of the specified file.

```
+ (id)imageRepWithContentsOfFile:(NSString *)filename
```

Parameters

filename

A full or relative pathname specifying the file to open. This string should include the filename extension.

Return Value
An initialized instance of an NSImageRep subclass, or nil if the image data could not be read.

Discussion
If sent to the NSImageRep class object, this method returns a newly allocated instance of a subclass of NSImageRep (chosen through the use of imageRepClassForFileType: (page 1404)) initialized with the contents of the specified file. If sent to a subclass of NSImageRep that recognizes the type of data in the file, it returns an instance of that subclass initialized with the contents of the file.

This method returns nil in any of the following cases:

- The message is sent to the NSImageRep class object and there are no subclasses in the NSImageRep class registry that handle the type of data in the specified file.
- The message is sent to a subclass of NSImageRep and that subclass cannot handle the type of data in the specified file.
- The NSImageRep subclass is unable to initialize itself with the contents of the specified file.

The NSImageRep subclass is initialized by creating an NSData object based on the contents of the file and passing it to the imageRepWithData: method. By default, the files handled include those with the extensions "tiff", "gif", "jpg", "pict", "pdf", and "eps".

Availability
Available in Mac OS X v10.0 and later.

See Also
+ imageFileTypes (page 1403)

Related Sample Code
PDFView

Declared In
NSImageRep.h

imageRepWithContentsOfURL:

Creates and returns an image representation object using the data at the specified URL

```
+ (id)imageRepWithContentsOfURL:(NSURL *)aURL
```

Parameters

aURL

The URL pointing to the image data.

Return Value
An initialized instance of an NSImageRep subclass, or nil if the image data could not be read.

Discussion
If sent to the NSImageRep class object, this method returns a newly allocated instance of a subclass of NSImageRep initialized with the contents of the specified URL. If sent to a subclass of NSImageRep that recognizes the data contained in the URL, it returns an instance of that subclass initialized with the data in the URL.

This method returns nil in any of the following cases:

- The message is sent to the NSImageRep class object and there are no subclasses in the NSImageRep class registry that handle the data contained in the specified URL.
- The message is sent to a subclass of NSImageRep and that subclass cannot handle the data contained in the specified URL.
- The NSImageRep subclass is unable to initialize itself with the contents of the specified URL.

The NSImageRep subclass is initialized by creating an NSData object based on the contents of the file, then passing it to the imageRepWithData: method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSImageRep.h

imageRepWithPasteboard:

Creates and returns an image representation object using the contents of the specified pasteboard.

```
+ (id)imageRepWithPasteboard:(NSPasteboard *)pasteboard
```

Parameters

pasteboard

The pasteboard containing the image data.

Return Value
An initialized instance of an NSImageRep subclass, or nil if the image data could not be read.

Discussion
If sent to the `NSImageRep` class object, this method returns a newly allocated instance of a subclass of `NSImageRep` initialized with the data in the specified pasteboard. If sent to a subclass of `NSImageRep` that recognizes the data on the pasteboard, it returns an instance of that subclass initialized with that data.

This method returns `nil` in any of the following cases:

- The message is sent to the `NSImageRep` class object and there are no subclasses in the `NSImageRep` class registry that handle data of the type contained in the specified pasteboard.
- The message is sent to a subclass of `NSImageRep` and that subclass cannot handle data of the type contained in the specified pasteboard.
- The `NSImageRep` subclass is unable to initialize itself with the contents of the pasteboard.

The `NSImageRep` subclass is initialized by creating an `NSData` object based on the data the specified pasteboard and passing it to the `imageRepWithData:` method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ imagePasteboardTypes` (page 1404)

Declared In
`NSImageRep.h`

imageTypes

Returns an array of UTI strings identifying the image types supported by the receiver, either directly or through a user-installed filter service.

```
+ (NSArray *)imageTypes
```

Return Value
An array of `NSString` objects, each of which contains a UTI identifying a supported image type. Some sample image-related UTI strings include "`public.image`", "`public.jpeg`", and "`public.tiff`". For a list of supported types, see `UTCoreTypes.h`.

Discussion
The returned list includes UTIs all file types supported by this image representation object plus those that can be opened by this image representation after being converted by a user-installed filter service. You can use the returned UTI strings with any method that supports UTIs.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSImageRep.h`

imageUnfilteredFileTypes

Returns the list of file types supported directly by the receiver.

```
+ (NSArray *)imageUnfilteredFileTypes
```

Return Value
An array of `NSString` objects. This array is empty by default. Subclasses must override to return the list of file formats they support.

Discussion
The returned file types can include encoded HFS file types as well as filename extensions. When creating a subclass of `NSImageRep`, override this method to return a list of strings representing the supported file types. For example, the `NSBitmapImageRep` class implements code similar to the following for this method:

```
+ (NSArray *)imageUnfilteredFileTypes {
    static NSArray *types = nil;

    if (!types) types = [[NSArray alloc]
        initWithObjects:@"tiff", @"gif", @"jpg",  @"bmp", nil];
    return types;
}
```

If your subclass supports the types supported by its superclass, you must explicitly get the array of types from the superclass and put them in the array returned by this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ imageFileTypes` (page 1403)
`+ imageUnfilteredFileTypes` (page 1333) (NSImage)

Declared In
`NSImageRep.h`

imageUnfilteredPasteboardTypes

Returns the list of pasteboard types supported directly by the receiver.

```
+ (NSArray *)imageUnfilteredPasteboardTypes
```

Return Value
An array of `NSString` objects. This array is empty by default. Subclasses must override to return the list of pasteboard formats they support.

Discussion
When creating a subclass of `NSImageRep`, override this method to return a list representing the supported pasteboard types. For example, the `NSBitmapImageRep` class implements code similar to the following for this method:

```
+ (NSArray *)imageUnfilteredPasteboardTypes {
    static NSArray *types = nil;

    if (!types) types = [[NSArray alloc] initWithObjects:NSTIFFPboardType,
nil];
    return types;
}
```

If your subclass supports the types supported by its superclass, you must explicitly get the list of types from the superclass and add them to the array returned by this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ imagePasteboardTypes` (page 1404)
`+ imageUnfilteredPasteboardTypes` (page 1333) (NSImage)

Declared In
`NSImageRep.h`

imageUnfilteredTypes

Returns an array of UTI strings identifying the image types supported directly by the receiver.

```
+ (NSArray *)imageUnfilteredTypes
```

Return Value
An array of `NSString` objects, each of which contains a UTI identifying a supported image type. Some sample image-related UTI strings include "`public.image`", "`public.jpeg`", and "`public.tiff`". For a list of supported types, see `UTCoreTypes.h`.

Discussion
The returned list includes UTI strings only for those file types that are supported directly by the receiver. It does not include types that are supported through user-installed filter services. You can use the returned UTI strings with any method that supports UTIs.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSImageRep.h`

registeredImageRepClasses

Returns an array containing the registered `NSImageRep` classes.

```
+ (NSArray *)registeredImageRepClasses
```

Return Value
An array of `Class` objects identifying the registered `NSImageRep` subclasses.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSImageRep.h`

registerImageRepClass:

Adds the specified class to the registry of available `NSImageRep` subclasses.

```
+ (void)registerImageRepClass:(Class)imageRepClass
```

Parameters

imageRepClass
The `Class` object for an `NSImageRep` subclass.

Discussion
This method posts an `NSImageRepRegistryDidChangeNotification` (page 1425), along with the receiving object, to the default notification center.

A good place to add image representation classes to the registry is in the `load` class method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ unregisterImageRepClass:` (page 1413)
`load` (NSObject)

Declared In
`NSImageRep.h`

unregisterImageRepClass:

Removes the specified `NSImageRep` subclass from the registry of available image representations.

```
+ (void)unregisterImageRepClass:(Class)imageRepClass
```

Parameters

imageRepClass
The `Class` object for an `NSImageRep` subclass.

Discussion
This method posts the `NSImageRepRegistryDidChangeNotification` (page 1425), along with the receiving object, to the default notification center.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ registerImageRepClass:` (page 1413)

Declared In
`NSImageRep.h`

Instance Methods

bitsPerSample

Returns the number of bits per sample in the receiver.

```
- (NSInteger)bitsPerSample
```

Return Value
The number of bits used to specify each component of data in a single pixel (for example, a value of 8 for an RGBA image means that each pixel is comprised of four 8-bit values). May also return `NSImageRepMatchesDevice` (page 1425).

Discussion
If the receiver is a planar image, this method returns the number of bits per sample per plane.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBitsPerSample:` (page 1421)
- `bitsPerPixel` (page 348) (NSBitmapImageRep)
- `samplesPerPixel` (page 364) (NSBitmapImageRep)
- `isPlanar` (page 363) (NSBitmapImageRep)

Related Sample Code
Quartz EB

Declared In
`NSImageRep.h`

CGImageForProposedRect:context:hints:

Returns a `CGImage` capturing the drawing of the receiver.

```
- (CGImageRef)CGImageForProposedRect:(NSRect *)proposedDestRect
    context:(NSGraphicsContext *)context hints:(NSDictionary *)hints
```

Parameters

`proposedDestRect`

On input, the proposed destination rectangle for drawing the image. If `NULL`, it defaults to the smallest pixel-integral rectangle containing {{0,0}, [self size]}. The *`proposedDestRect`* is in user space in the reference context.

On output the proposedDestRect may have been altered. This is because a `CGImage` is necessarily pixel-integral, while an `NSImage` is not. In order to produce a `CGImage` for rect `(0.5, 0.5, 4.0, 4.0)` without distortion or double-antialiasing, we may have to produce a 5x5 `CGImage`, and also inflate the *`proposedDestRect`*. Drawing the `CGImage` in the out-value *`proposedDestRect`* is the same as drawing the NSImage in the in-value of proposed rect.

`context`

A graphics context., Can be `NULL`.

hints

An optional dictionary of hints that provide more context for selecting or generating the image. See `Image_Hint_Dictionary_Keys` (page 1375) for a summary of the possible key-value pairs.

Return Value
A `CGImageRef`. This may be an existing `CGImage` if one is available. If not, a new `CGImage` is created.

Discussion
An `NSImage` is potentially resolution independent, and may have representations that allow it to draw well in many contexts. A `CGImage` is more like a single pixel-based representation. This method produces a snapshot of how the `NSImage` would draw if it was asked to draw in the proposed rectangle in the graphics context.

All input parameters are optional. They provide hints for how to choose among existing CGImages, or how to create one if there isn't already a CGImage available. The parameters are only hints.

This method is intended as an override point for image rep subclasses that naturally have a `CGImage` available. For example, `NSBitmapImageRep` overrides it to return the `CGImage` that naturally backs the rep. You don't need to override the method except possibly for performance, though. The NSImageRep-level implementation will produce a CGImage by making a buffer and calling `[self draw]`. That's likely to be the best possible implementation for reps that aren't naturally CGImage backed. The `draw` (page 1415) remains the only method of NSImageRep that a subclasser really needs to override.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSImageRep.h`

colorSpaceName

Returns the name of the receiver's color space.

```
- (NSString *)colorSpaceName
```

Return Value
The colorspace name, or `NSCalibratedRGBColorSpace` if no name has been assigned.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setColorSpaceName:` (page 1421)

Declared In
`NSImageRep.h`

draw

Implemented by subclasses to draw the image in the current coordinate system.

```
- (BOOL)draw
```

Return Value
`YES` if the image was successfully drawn; otherwise, `NO` if there was a problem. The default version of this method simply returns `YES`.

Discussion
Subclass override this method to draw the image using the image data. By the time this method is called, the graphics state is already configured for you to draw the image at location (0.0, 0.0) in the current coordinate system.

The standard Application Kit subclasses all draw the image using the `NSCompositeCopy` composite operation defined in the "Constants" (page 1375) section of `NSImage`. Using the copy operator, the image data overwrites the destination without any blending effects. Transparent (alpha) regions in the source image appear black. To use other composite operations, you must place the representation into an `NSImage` object and use its `drawAtPoint:fromRect:operation:fraction:` (page 1346) or `drawInRect:fromRect:operation:fraction:` (page 1347) methods.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
GLUT

Declared In
`NSImageRep.h`

drawAtPoint:

Draws the receiver's image data at the specified point in the current coordinate system.

```
- (BOOL)drawAtPoint:(NSPoint)aPoint
```

Parameters
aPoint
The point in the current coordinate system at which to draw the image.

Return Value
`YES` if the image was successfully drawn; otherwise, `NO`. If the size of the image has not yet been set, this method returns `NO` immediately

Discussion
This method sets the origin of the current coordinate system to the specified point and then invokes the receiver's `draw` method to draw the image at that point. Upon completion, it restores the current coordinates to their original setting. If *aPoint* is (0.0, 0.0), this method simply invokes the `draw` (page 1415) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setSize:` (page 1423)
- `drawInRect:` (page 1417)
- `drawInRect:fromRect:operation:fraction:respectFlipped:hints:` (page 1417)

Declared In
`NSImageRep.h`

drawInRect:

Draws the image, scaling it (as needed) to fit the specified rectangle.

```
- (BOOL)drawInRect:(NSRect)rect
```

Parameters

rect

The rectangle in the current coordinate system in which to draw the image.

Return Value

`YES` if the image was successfully drawn; otherwise, `NO`. If the size of the image has not yet been set, this method returns `NO` immediately

Discussion

This method sets the origin of the current coordinate system to the origin of the specified rectangle before invoking the receiver's `draw` (page 1415) method. If the rectangle size is different from the image's native size, this method adjusts the coordinate transform, causing the image to be scaled appropriately. After the `draw` method returns, the coordinate system changes are undone, restoring the original graphics state.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setSize:` (page 1423)
- `drawAtPoint:` (page 1416)
- `drawInRect:fromRect:operation:fraction:respectFlipped:hints:` (page 1417)

Related Sample Code

PDF Annotation Editor

PDFView

Sketch-112

Declared In

`NSImageRep.h`

drawInRect:fromRect:operation:fraction:respectFlipped:hints:

Draws all or part of the image in the specified rectangle in the current coordinate system.

```
- (BOOL)drawInRect:(NSRect)dstSpacePortionRect fromRect:(NSRect)srcSpacePortionRect
    operation:(NSCompositingOperation)op fraction:(CGFloat)requestedAlpha
  respectFlipped:(BOOL)respectContextIsFlipped hints:(NSDictionary *)hints
```

Parameters

dstSpacePortionRect

The rectangle in which to draw the image, specified in the current coordinate system.

srcSpacePortionRect

The source rectangle specifying the portion of the image you want to draw. The coordinates of this rectangle must be specified using the image's own coordinate system. If you pass in `NSZeroRect`, the entire image is drawn.

op

The compositing operation to use when drawing the image. See the `NSCompositingOperation` (page 1375) constants.

requestedAlpha

The opacity of the image, specified as a value from 0.0 to 1.0. Specifying a value of 0.0 draws the image as fully transparent while a value of 1.0 draws the image as fully opaque. Values greater than 1.0 are interpreted as 1.0.

respectContextIsFlipped

`YES` if the flipped context of the receiver should be respected, otherwise `NO`.

hints

An optional dictionary of hints that provide more context for selecting or generating the image. See `Image_Hint_Dictionary_Keys` (page 1375) for a summary of the possible key-value pairs.

Return Value
`YES` if the image was successfully drawn; otherwise, `NO`.

Discussion
If the *srcSpacePortionRect* and *dstSpacePortionRect* rectangles have different sizes, the source portion of the image is scaled to fit the specified destination rectangle.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `draw` (page 1415)
- `drawAtPoint:` (page 1416)
- `drawInRect:` (page 1417)

Declared In
`NSImageRep.h`

hasAlpha

Returns a Boolean value indicating whether the receiver has an alpha channel.

```
- (BOOL)hasAlpha
```

Return Value
`YES` if the receiver has a known alpha channel; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlpha:` (page 1420)

Related Sample Code
LayerBackedOpenGLView
NSOpenGL Fullscreen

Declared In
`NSImageRep.h`

isOpaque

Returns a Boolean value indicating whether the receiver is opaque.

```
- (BOOL)isOpaque
```

Return Value
`YES` if the receiver is opaque; otherwise, `NO`.

Discussion
Use this method to test whether an image representation completely covers the area within the rectangle returned by the `size` (page 1424) method.

The returned value does not indicate whether the image has an alpha channel or if there is partial or complete transparency when drawing the image rep. Use the `hasAlpha` (page 1418) method to determine if the image has an alpha channel.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setOpaque:` (page 1422)

Declared In
`NSImageRep.h`

pixelsHigh

Returns the height of the image, measured in pixels.

```
- (NSInteger)pixelsHigh
```

Return Value
The height of the image, measured in the units of the device coordinate space. This value is usually derived from the image data itself.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setPixelsHigh:` (page 1422)
`- pixelsWide` (page 1420)
`- size` (page 1424)
`NSImageRepMatchesDevice` (page 1425)

Related Sample Code
Cocoa OpenGL
GLUT
OpenCL_OceanWave
Quartz EB
Reducer

Declared In
NSImageRep.h

pixelsWide

Returns the width of the image, measured in pixels.

```
- (NSInteger)pixelsWide
```

Return Value
The width of the image, measured in the units of the device coordinate space. This value is usually derived from the image data itself.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setPixelsWide: (page 1423)
- pixelsHigh (page 1419)
- size (page 1424)

NSImageRepMatchesDevice (page 1425)

Related Sample Code
Cocoa OpenGL
GLUT
OpenCL_OceanWave
Quartz EB
Reducer

Declared In
NSImageRep.h

setAlpha:

Informs the receiver that its image data has an alpha component.

```
- (void)setAlpha:(BOOL)flag
```

Parameters
flag
YES if you want the receiver to have an alpha component; otherwise NO.

Discussion
Subclasses should call this method when loading image data to notify the parent class whether that data contains an alpha component. Passing in a value of YES does not add an alpha channel to the image data itself; it merely records the fact that the data has an alpha channel.

Availability
Available in Mac OS X v10.0 and later.

See Also
- hasAlpha (page 1418)

Declared In
NSImageRep.h

setBitsPerSample:

Informs the receiver that its image data has the specified number of bits for each component of a pixel.

```
- (void)setBitsPerSample:(NSInteger)anInt
```

Parameters

anInt
The number of bits used by each component of a pixel, or NSImageRepMatchesDevice (page 1425).

Discussion
Subclasses should call this method when loading image data to notify the parent class of how many bits each sample uses. Specifying a value that differs from the actual image data does not change the bit depth of the image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- bitsPerSample (page 1414)

Declared In
NSImageRep.h

setColorSpaceName:

Informs the receiver of the color space used by the image data.

```
- (void)setColorSpaceName:(NSString *)string
```

Parameters

string
The name of the color space used by the image data.

Discussion
By default, an NSImageRep object's color space name is NSCalibratedRGBColorSpace. Color space names are defined as part of the NSColor class, in NSGraphics.h. The following are valid color space names:

```
NSCalibratedWhiteColorSpace
NSCalibratedBlackColorSpace
NSCalibratedRGBColorSpace
NSDeviceWhiteColorSpace
NSDeviceBlackColorSpace
NSDeviceRGBColorSpace
NSDeviceCMYKColorSpace
NSNamedColorSpace
NSCustomColorSpace
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `colorSpaceName` (page 1415)

Declared In
`NSImageRep.h`

setOpaque:

Sets whether the receiver's image is opaque.

```
- (void)setOpaque:(BOOL)flag
```

Parameters

flag

`YES` if the image should be treated as fully opaque; otherwise, `NO` to indicate the image may include some transparent regions.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isOpaque` (page 1419)

Declared In
`NSImageRep.h`

setPixelsHigh:

Informs the receiver of the image data height.

```
- (void)setPixelsHigh:(NSInteger)anInt
```

Parameters

anInt

The height of the image, measured in pixels.

Discussion
Subclasses should call this method when loading image data to notify the parent class of the image height. You cannot use this method to change the actual number of pixels in the image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `pixelsHigh` (page 1419)
- `setPixelsWide:` (page 1423)
- `setSize:` (page 1423)
- `NSImageRepMatchesDevice` (page 1425)

Declared In
NSImageRep.h

setPixelsWide:

Informs the receiver of the image data width.

```
- (void)setPixelsWide:(NSInteger)anInt
```

Parameters

anInt

The width of the image, measured in pixels.

Discussion
Subclasses should call this method when loading image data to notify the parent class of the image width. You cannot use this method to change the actual number of pixels in the image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- pixelsWide (page 1420)
- setPixelsHigh: (page 1422)
- setSize: (page 1423)

NSImageRepMatchesDevice (page 1425)

Declared In
NSImageRep.h

setSize:

Sets the size of the image representation to the specified value.

```
- (void)setSize:(NSSize)aSize
```

Parameters

aSize

The new size of the image representation, measured in points in the user coordinate space.

Discussion
This method determines the size of the image when it's rendered. It is not necessarily the same as the width and height of the image in pixels as specified by the image data, nor must it be equal to the size set for the NSImage object that wraps this image representation. You must set the image size before you can render it.

The size of an image representation combined with the physical dimensions of the image data determine the resolution of the image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- size (page 1424)

- draw (page 1415)
- setPixelsHigh: (page 1422)
- setPixelsWide: (page 1423)

Related Sample Code
SimpleImageFilter

Declared In
NSImageRep.h

size

Returns the size of the image representation.

```
- (NSSize)size
```

Return Value
The size of the image representation, measured in points in the user coordinate space.

Discussion
This size is the size of the image representation when it's rendered. It is not necessarily the same as the width and height of the image in pixels as specified by the image data, nor must it be equal to the size set for the NSImage object that wraps this image representation.

The size of an image representation combined with the physical dimensions of the image data determine the resolution of the image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setSize: (page 1423)
- pixelsHigh (page 1419)
- pixelsWide (page 1420)

Related Sample Code
NURBSSurfaceVertexProg
StillMotion
SurfaceVertexProgram
Vertex Optimization
VertexPerformanceDemo

Declared In
NSImageRep.h

Constants

Display Device Matching

The following constant is used by `NSImageRep` to denote an attribute whose value changes to match the display device.

```
enum {
   NSImageRepMatchesDevice
};
```

Constants

`NSImageRepMatchesDevice`

Indicates that the value of certain attributes, such as the number of colors or bits per sample, will change to match the display device.

This value can be passed in (or received back) as the value of `bitsPerSample` (page 1414), `pixelsWide` (page 1420), and `pixelsHigh` (page 1419).

Available in Mac OS X v10.0 and later.

Declared in `NSImageRep.h`.

Deprecated Notification Name

The following constant maps to the new notification and is for legacy code only.

```
#define NSImageRepRegistryChangedNotification NSImageRepRegistryDidChangeNotification
```

Constants

`NSImageRepRegistryChangedNotification`

An older name for the `NSImageRepRegistryDidChangeNotification` (page 1425) notification. Do not use.

Available in Mac OS X v10.0 and later.

Declared in `NSImageRep.h`.

Notifications

NSImageRepRegistryDidChangeNotification

Posted whenever the `NSImageRep` class registry changes.

The notification object is the image class that is registered or unregistered. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSImageRep.h`

NSImageView Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSImageView.h
Companion guide	Image Views
Related sample code	From A View to A Movie GLUT ImageClient MyPhoto PDF Calendar

Overview

An `NSImageView` object displays a single image from an `NSImage` object in a frame and can optionally allow a user to drag an image to it.

Tasks

Choosing the Image

- `image` (page 1429)
  Returns the `NSImage` object displayed by the receiver.
- `setImage:` (page 1432)
  Sets the image of the receiver.

Choosing the Frame

- `imageFrameStyle` (page 1430)
  Returns the style of frame that appears around the image.
- `setImageFrameStyle:` (page 1434)
  Sets the kind of frame that borders the image.

Aligning and Scaling the Image

- `imageAlignment` (page 1430)
  Returns the position of the cell's image in the frame.
- `setImageAlignment:` (page 1433)
  Sets the position of the image in the frame.
- `imageScaling` (page 1431)
  Returns the way the cell's image alters to fit the frame.
- `setImageScaling:` (page 1434)
  Sets the way the image alters to fit the frame.

Responding to User Events

- `isEditable` (page 1431)
  Returns a Boolean value indicating whether the user can drag a new image into the frame.
- `setEditable:` (page 1432)
  Sets whether the user can drag a new image into the frame.

Animating Image Playback

- `animates` (page 1429)
  Returns a Boolean value indicating whether the receiver automatically plays animated images.
- `setAnimates:` (page 1432)
  Sets whether the receiver automatically plays an animated image that is assigned to it.

Pasteboard Support

- `setAllowsCutCopyPaste:` (page 1431)
  Sets whether the receiver allows the user to cut, copy and paste the image contents.
- `allowsCutCopyPaste` (page 1429)
  Returns a Boolean value indicating whether the receiver allows the user to cut, copy and paste of the image contents.

Instance Methods

allowsCutCopyPaste

Returns a Boolean value indicating whether the receiver allows the user to cut, copy and paste of the image contents.

```
- (BOOL)allowsCutCopyPaste
```

Return Value
`YES` if the user can cut, copy, and paste the image contents; otherwise, `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAllowsCutCopyPaste:` (page 1431)

Declared In
`NSImageView.h`

animates

Returns a Boolean value indicating whether the receiver automatically plays animated images.

```
- (BOOL)animates
```

Return Value
`YES` if the receiver automatically plays animated images; otherwise, `NO`. The default value is `YES` for `NSImageView` objects you create programmatically. For `NSImageView` objects loaded from a nib file, the control takes the value set in Interface Builder.

Discussion
The timing and looping characteristics of the animation are taken from the image data. If this method returns `NO`, the receiver displays the first frame of the animation only.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setAnimates:` (page 1432)

Declared In
`NSImageView.h`

image

Returns the `NSImage` object displayed by the receiver.

```
- (NSImage *)image
```

Return Value
The `NSImage` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setImage:` (page 1432)

Related Sample Code
CoreImageGLTextureFBO
Cropped Image
DockTile
PDF Calendar
Reducer

Declared In
`NSImageView.h`

imageAlignment

Returns the position of the cell's image in the frame.

```
- (NSImageAlignment)imageAlignment
```

Return Value
The image alignment. For a list of possible alignments, see `setImageAlignment:` (page 1433). The default value is `NSImageAlignCenter`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSImageView.h`

imageFrameStyle

Returns the style of frame that appears around the image.

```
- (NSImageFrameStyle)imageFrameStyle
```

Return Value
The current image style. For a list of frame styles, see `setImageFrameStyle:` (page 1434). The default value is `NSImageFrameNone`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Transformed Image

Declared In
`NSImageView.h`

imageScaling

Returns the way the cell's image alters to fit the frame.

```
- (NSImageScaling)imageScaling
```

Return Value
The scaling option. For a list of possible values, see `setImageScaling:` (page 1434). The default value is `NSScaleProportionally`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaSlides

Declared In
`NSImageView.h`

isEditable

Returns a Boolean value indicating whether the user can drag a new image into the frame.

```
- (BOOL)isEditable
```

Return Value
`YES` if the user can drag an image into the receiver's frame; otherwise, `NO`. The default value is `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setEditable:` (page 1432)

Declared In
`NSImageView.h`

setAllowsCutCopyPaste:

Sets whether the receiver allows the user to cut, copy and paste the image contents.

```
- (void)setAllowsCutCopyPaste:(BOOL)allow
```

Parameters

allow

`YES` if the user can cut, copy, and paste the image contents; otherwise, `NO` to prevent the use of pasteboard operations.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- allowsCutCopyPaste` (page 1429)

Declared In
NSImageView.h

setAnimates:

Sets whether the receiver automatically plays an animated image that is assigned to it.

```
- (void)setAnimates:(BOOL)flag
```

Parameters

flag

YES if the receiver should automatically plays animated images; otherwise, NO.

Discussion
The timing and looping characteristics of the animation are taken from the image data. If you specify NO, the receiver displays the first frame of the animation only.

Availability
Available in Mac OS X v10.3 and later.

See Also
- animates (page 1429)

Declared In
NSImageView.h

setEditable:

Sets whether the user can drag a new image into the frame.

```
- (void)setEditable:(BOOL)flag
```

Parameters

flag

YES if the user can drag an image into the receiver's frame; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isEditable (page 1431)

Related Sample Code
FunHouse
GLUT

Declared In
NSImageView.h

setImage:

Sets the image of the receiver.

```
- (void)setImage:(NSImage *)image
```

Parameters

image

The image to display in the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- image (page 1429)

Related Sample Code
DockTile
FunHouse
GLUT
ImageClient
QTMetadataEditor

Declared In
NSImageView.h

setImageAlignment:

Sets the position of the image in the frame.

```
- (void)setImageAlignment:(NSImageAlignment)alignment
```

Parameters

alignment

The possible values for this parameter are:

- NSImageAlignLeft
- NSImageAlignRight
- NSImageAlignCenter
- NSImageAlignTop
- NSImageAlignBottom
- NSImageAlignTopLeft
- NSImageAlignTopRight
- NSImageAlignBottomLeft
- NSImageAlignBottomRight

Availability
Available in Mac OS X v10.0 and later.

See Also
- imageAlignment (page 1430)

Related Sample Code
GLUT

Declared In
`NSImageView.h`

setImageFrameStyle:

Sets the kind of frame that borders the image.

```
- (void)setImageFrameStyle:(NSImageFrameStyle)frameStyle
```

Parameters

frameStyle

The possible values for this parameter are as follows:

- `NSImageFrameNone`—an invisible frame
- `NSImageFramePhoto`—a thin black outline and a dropped shadow
- `NSImageFrameGrayBezel`—a gray, concave bezel that makes the image look sunken
- `NSImageFrameGroove`—a thin groove that looks etched around the image
- `NSImageFrameButton`—a convex bezel that makes the image stand out in relief, like a button

Availability
Available in Mac OS X v10.0 and later.

See Also
- `imageFrameStyle` (page 1430)

Related Sample Code
FunHouse
Transformed Image

Declared In
`NSImageView.h`

setImageScaling:

Sets the way the image alters to fit the frame.

```
- (void)setImageScaling:(NSImageScaling)scaling
```

Parameters

scaling

The possible values for this parameter are:

- `NSScaleProportionally`. If the image is too large, it shrinks to fit inside the frame. The proportions of the image are preserved. The image is never scaled up to fit a larger frame.
- `NSScaleToFit`. The image shrinks or expands, and its proportions distort, until it exactly fits the frame.
- `NSScaleNone`. The size and proportions of the image don't change. If the frame is too small to display the whole image, the edges of the image are trimmed off.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- imageScaling` (page 1431)

Related Sample Code

FunHouse

Declared In

`NSImageView.h`

NSLayoutManager Class Reference

Inherits from	NSObject
Conforms to	NSGlyphStorage NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSLayoutManager.h
Companion guides	Text System Overview Text Layout Programming Guide
Related sample code	Quartz Composer WWDC 2005 TextEdit QuickLookSketch Sketch+Accessibility Sketch-112 TextSizingExample

Overview

An `NSLayoutManager` object coordinates the layout and display of characters held in an `NSTextStorage` object. It maps Unicode character codes to glyphs, sets the glyphs in a series of `NSTextContainer` objects, and displays them in a series of `NSTextView` objects. In addition to its core function of laying out text, an `NSLayoutManager` object coordinates its `NSTextView` objects, provides services to those text views to support `NSRulerView` instances for editing paragraph styles, and handles the layout and display of text attributes not inherent in glyphs (such as underline or strikethrough). You can create a subclass of `NSLayoutManager` to handle additional text attributes, whether inherent or not.

Text Antialiasing

`NSLayoutManager` provides the threshold for text antialiasing. It looks at the `AppleAntiAliasingThreshold` default value. If the font size is smaller than or equal to this threshold size, the text is rendered aliased by `NSLayoutManager`. You can change the threshold value from the Appearance pane of System Preferences.

Thread Safety of NSLayoutManager

Generally speaking, a given layout manager (and associated objects) should not be used in more than one block, operation, or thread at a time. Most layout managers are used on the main thread, since it is the main thread on which their text views are displayed, and since background layout occurs on the main thread. If it is intended that a layout manager should be used on a background thread, first make sure that text views associated with that layout manager (if any) are not displayed while the layout manager is being used on the background thread, and, second, turn off background layout for that layout manager while it is being used on the background thread.

Noncontiguous Layout

Noncontiguous layout is an optional layout manager behavior new in Mac OS X v10.5. Previously, both glyph generation and layout were always performed, in order, from the beginning to the end of the document. When noncontiguous layout is turned on, however, the layout manager gains the option of performing glyph generation or layout for one portion of the document without having done so for previous sections. This can provide significant performance improvements for large documents.

Noncontiguous layout is not turned on automatically because direct clients of `NSLayoutManager` typically have relied on the previous behavior—for example, by forcing layout for a given glyph range, and then assuming that previous glyphs would therefore be laid out. Clients who use `NSLayoutManager` only indirectly—for example, those who use `NSTextView` without directly calling the underlying layout manager—can usually turn on noncontiguous layout without difficulty. Clients using `NSLayoutManager` directly need to examine their usage before turning on noncontiguous layout.

To turn on noncontiguous layout, use `setAllowsNonContiguousLayout:` (page 1498). In addition, see the other methods in "Managing Noncontiguous Layout" (page 1447), many of which enable you to ensure that glyph generation and layout are performed for specified portions of the text. The behavior of a number of other layout manager methods is affected by the state of noncontiguous layout, as noted in the discussion sections of those method descriptions.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

NSGlyphStorage
- `- attributedString` (page 3688)
- `- insertGlyphs:length:forStartingGlyphAtIndex:characterIndex:` (page 3688)
- `- layoutOptions` (page 3689)
- `- setIntAttribute:value:forGlyphAtIndex:` (page 3689)

Tasks

Initializing

- `init` (page 1478)
  Initializes the receiver, a newly created `NSLayoutManager` object.

Setting the Text Storage

- `setTextStorage:` (page 1509)
  Sets the receiver's `NSTextStorage` object.
- `textStorage` (page 1520)
  Returns the receiver's text storage object.
- `attributedString` (page 1451)
  Returns the text storage object from which the `NSGlyphGenerator` object procures characters for glyph generation.
- `replaceTextStorage:` (page 1496)
  Replaces the `NSTextStorage` object for the group of text-system objects containing the receiver with the given text storage object.

Setting Text Containers

- `textContainers` (page 1520)
  Returns the receiver's text containers.
- `addTextContainer:` (page 1449)
  Appends the given text container to the series of text containers where the receiver arranges text.
- `insertTextContainer:atIndex:` (page 1480)
  Inserts the given text container into the series of text containers at the given index.
- `removeTextContainerAtIndex:` (page 1495)
  Removes the text container at the given index and invalidates the layout as needed.

Setting the Glyph Generator

- `setGlyphGenerator:` (page 1502)
  Sets the glyph generator used by this layout manager.
- `glyphGenerator` (page 1472)
  Returns the glyph generator used by this layout manager.

Invalidating Glyphs and Layout

- `invalidateGlyphsForCharacterRange:changeInLength:actualCharacterRange:` (page 1482)
  Invalidates the cached glyphs for the characters in the given character range, adjusts the character indices of all the subsequent glyphs by the change in length, and invalidates the new character range.
- `invalidateGlyphsOnLayoutInvalidationForGlyphRange:` (page 1483)
  Specifies explicitly when portions of the glyph stream depend on layout.
- `invalidateLayoutForCharacterRange:isSoft:actualCharacterRange:` (page 1483)
  Invalidates the layout information for the glyphs mapped to the given range of characters.
- `invalidateLayoutForCharacterRange:actualCharacterRange:` (page 1483)
  Invalidates the layout information for the glyphs mapped to the given range of characters.
- `invalidateDisplayForCharacterRange:` (page 1481)
  Invalidates display for the given character range.
- `invalidateDisplayForGlyphRange:` (page 1482)
  Marks the glyphs in the given glyph range as needing display, as well as the appropriate regions of the `NSTextView` objects that display those glyphs (using the `NSView` method `setNeedsDisplayInRect:` (page 3225)).
- `textContainerChangedGeometry:` (page 1518)
  Invalidates the layout information, and possibly glyphs, for the given text container and all subsequent `NSTextContainer` objects.
- `textContainerChangedTextView:` (page 1518)
  Updates information needed to manage `NSTextView` objects in the given text container.
- `textStorage:edited:range:changeInLength:invalidatedRange:` (page 1521)
  Invalidates glyph and layout information for a portion of the text in the given text storage object.

Enabling Background Layout

- `setBackgroundLayoutEnabled:` (page 1499)
  Specifies whether the receiver generates glyphs and lays them out when the application's run loop is idle.
- `backgroundLayoutEnabled` (page 1451)
  Indicates whether the receiver generates glyphs and lays out text when the application's run loop is idle.

Accessing Glyphs

- `insertGlyph:atGlyphIndex:characterIndex:` (page 1479)
  Inserts a single glyph into the glyph stream at the given index and maps it to the character at the given character index.
- `insertGlyphs:length:forStartingGlyphAtIndex:characterIndex:` (page 1479)
  Inserts the given glyphs into the glyph cache at the given index and maps them to characters beginning at the given character index.
- `isValidGlyphIndex:` (page 1484)
  Indicates whether the specified index refers to a valid glyph, otherwise `NO`.

- `glyphAtIndex:` (page 1471)
  Returns the glyph at *glyphIndex*.
- `glyphAtIndex:isValidIndex:` (page 1472)
  If the given index is valid, returns the glyph at that location and optionally returns a flag indicating whether the requested index is in range.
- `replaceGlyphAtIndex:withGlyph:` (page 1495)
  Replaces the glyph at the given index with a new glyph.
- `getGlyphs:range:` (page 1468)
  Fills the passed-in buffer with a sequence of glyphs
- `getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:` (page 1469)
  Returns the glyphs and information needed to perform layout for the given glyph range.
- `getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:bidiLevels:` (page 1469)
  Returns the glyphs and information needed to perform layout for the given glyph range.
- `deleteGlyphsInRange:` (page 1457)
  Deletes the glyphs in the given range from the receiver's glyph store.
- `numberOfGlyphs` (page 1491)
  Returns the number of glyphs in the receiver.

Mapping Characters to Glyphs

- `setCharacterIndex:forGlyphAtIndex:` (page 1500)
  Sets the index of the character corresponding to the glyph at the given glyph index.
- `characterIndexForGlyphAtIndex:` (page 1453)
  Returns the index in the text storage for the first character associated with the given glyph.
- `glyphIndexForCharacterAtIndex:` (page 1473)
  Returns the index of the first glyph associated with the character at the specified index.
- `characterRangeForGlyphRange:actualGlyphRange:` (page 1455)
  Returns the range of characters that generated the glyphs in the given glyph range.
- `glyphRangeForCharacterRange:actualCharacterRange:` (page 1476)
  Returns the range of glyphs that are generated from the characters in the given character range.

Setting Glyph Attributes

- `intAttribute:forGlyphAtIndex:` (page 1481)
  Returns the value of the attribute identified by the given attribute tag for the glyph at the given index.
- `setIntAttribute:value:forGlyphAtIndex:` (page 1503)
  Sets a custom attribute value for a given glyph.
- `setAttachmentSize:forGlyphRange:` (page 1498)
  Sets the size at which the given glyph (assumed to be an attachment) is asked to draw in the given glyph range.
- `attachmentSizeForGlyphAtIndex:` (page 1450)
  For a glyph corresponding to an attachment, returns the size for the attachment cell to occupy.

- `setDefaultAttachmentScaling:` (page 1500)
  Sets the default scaling behavior to the given scaling if an attachment image is too large to fit in a text container.
- `defaultAttachmentScaling` (page 1456)
  Returns the default behavior desired if an attachment image is too large to fit in a text container.
- `showAttachmentCell:inRect:characterIndex:` (page 1512)
  Draws an attachment cell.

Handling Layout for Text Containers

- `setTextContainer:forGlyphRange:` (page 1509)
  Sets text container where the glyphs in the given range are laid out.
- `glyphRangeForTextContainer:` (page 1477)
  Returns the range of glyphs laid out within the given text container.
- `textContainerForGlyphAtIndex:effectiveRange:` (page 1519)
  Returns the container in which the given glyph is laid out and (optionally) by reference the whole range of glyphs that are in that container.
- `textContainerForGlyphAtIndex:effectiveRange:withoutAdditionalLayout:` (page 1519)
  Returns the container in which the given glyph is laid out and (optionally) by reference the whole range of glyphs that are in that container.
- `usedRectForTextContainer:` (page 1524)
  Returns the bounding rectangle for the glyphs laid out in the given text container.
- `characterIndexForPoint:inTextContainer:fractionOfDistanceBetweenInsertionPoints:` (page 1454)
  Returns the index of the character falling under the given point, expressed in the given container's coordinate system.

Handling Line Fragment Rectangles

- `setLineFragmentRect:forGlyphRange:usedRect:` (page 1505)
  Associates the given line fragment bounds with the given range of glyphs.
- `lineFragmentRectForGlyphAtIndex:effectiveRange:` (page 1487)
  Returns the rectangle for the line fragment in which the given glyph is laid out and (optionally), by reference, the whole range of glyphs that are in that fragment.
- `lineFragmentRectForGlyphAtIndex:effectiveRange:withoutAdditionalLayout:` (page 1487)
  Returns the line fragment rectangle containing the glyph at the given glyph index.
- `lineFragmentUsedRectForGlyphAtIndex:effectiveRange:` (page 1488)
  Returns the usage rectangle for the line fragment in which the given glyph is laid and (optionally) by reference the whole range of glyphs that are in that fragment.
- `lineFragmentUsedRectForGlyphAtIndex:effectiveRange:withoutAdditionalLayout:` (page 1489)
  Returns the usage rectangle for the line fragment in which the given glyph is laid and (optionally) by reference the whole range of glyphs that are in that fragment.
- `setExtraLineFragmentRect:usedRect:textContainer:` (page 1502)
  Sets the bounds and container for the extra line fragment.

- `extraLineFragmentRect` (page 1464)
  Returns the rectangle defining the extra line fragment for the insertion point at the end of a text (either in an empty text or after a final paragraph separator).
- `extraLineFragmentUsedRect` (page 1465)
  Returns the rectangle enclosing the insertion point drawn in the extra line fragment rectangle.
- `extraLineFragmentTextContainer` (page 1465)
  Returns the text container that contains the extra line fragment rectangle.
- `setDrawsOutsideLineFragment:forGlyphAtIndex:` (page 1501)
  Specifies whether the given glyph exceeds the bounds of the line fragment where it's laid out.
- `drawsOutsideLineFragmentForGlyphAtIndex:` (page 1459)
  Indicates whether the glyph draws outside of its line fragment rectangle.

Laying Out Glyphs

- `setLocation:forStartOfGlyphRange:` (page 1506)
  Sets the location for the first glyph of the given range.
- `setLocations:startingGlyphIndexes:count:forGlyphRange:` (page 1506)
  Sets locations for many glyph ranges at once.
- `locationForGlyphAtIndex:` (page 1490)
  Returns the location for the given glyph within its line fragment.
- `rangeOfNominallySpacedGlyphsContainingIndex:` (page 1491)
  Returns the range for the glyphs around the given glyph that can be displayed using only their advancements from the font, without pairwise kerning or other adjustments to spacing.
- `getLineFragmentInsertionPointsForCharacterAtIndex:alternatePositions:inDisplayOrder:positions:characterIndexes:` (page 1470)
  Returns insertion points in bulk for a given line fragment.
- `rectArrayForCharacterRange:withinSelectedCharacterRange:inTextContainer:rectCount:` (page 1492)
  Returns an array of rectangles and, by reference, the number of such rectangles, that define the region in the given container enclosing the given character range.
- `rectArrayForGlyphRange:withinSelectedGlyphRange:inTextContainer:rectCount:` (page 1493)
  Returns an array of rectangles and, by reference, the number of such rectangles, that define the region in the given container enclosing the given glyph range.
- `boundingRectForGlyphRange:inTextContainer:` (page 1452)
  Returns a single bounding rectangle (in container coordinates) enclosing all glyphs and other marks drawn in the given text container for the given glyph range, including glyphs that draw outside their line fragment rectangles and text attributes such as underlining.
- `glyphRangeForBoundingRect:inTextContainer:` (page 1475)
  Returns the smallest contiguous range for glyphs that are laid out wholly or partially within the given rectangle in the given text container.
- `glyphRangeForBoundingRectWithoutAdditionalLayout:inTextContainer:` (page 1475)
  Returns the smallest contiguous range for glyphs that are laid out wholly or partially within the given rectangle in the given text container.

- `glyphIndexForPoint:inTextContainer:fractionOfDistanceThroughGlyph:` (page 1474)
  Returns the index of the glyph falling under the given point, expressed in the given container's coordinate system.
- `fractionOfDistanceThroughGlyphForPoint:inTextContainer:` (page 1468)
  This method is a primitive for `glyphIndexForPoint:inTextContainer:fractionOfDistanceThroughGlyph:` (page 1474). You should always call the main method, not the primitives.
- `glyphIndexForPoint:inTextContainer:` (page 1473)
  This method is a primitive for `glyphIndexForPoint:inTextContainer:fractionOfDistanceThroughGlyph:` (page 1474). You should always call the main method, not the primitives.

Handling Layout for Text Blocks

- `setLayoutRect:forTextBlock:glyphRange:` (page 1504)
  Sets the layout rectangle enclosing the given text block containing the given glyph range.
- `layoutRectForTextBlock:glyphRange:` (page 1486)
  Returns the layout rectangle within which the given text block containing the given glyph range is to be laid out.
- `setBoundsRect:forTextBlock:glyphRange:` (page 1499)
  Sets the bounding rectangle enclosing a given text block containing the given glyph range.
- `boundsRectForTextBlock:glyphRange:` (page 1453)
  Returns the bounding rectangle enclosing the given text block containing the given glyph range.
- `layoutRectForTextBlock:atIndex:effectiveRange:` (page 1486)
  Returns the layout rectangle within which the given text block containing the glyph at the given index is to be laid out.
- `boundsRectForTextBlock:atIndex:effectiveRange:` (page 1452)
  Returns the bounding rectangle within which the given text block containing the glyph at the given index is to be laid out.

Displaying Special Glyphs

- `setNotShownAttribute:forGlyphAtIndex:` (page 1507)
  Sets the glyph at the given index to be one that isn't shown.
- `notShownAttributeForGlyphAtIndex:` (page 1490)
  Indicates whether the glyph at the given index is one that isn't shown.
- `setShowsInvisibleCharacters:` (page 1508)
  Specifies whether to substitute visible glyphs for whitespace and other typically invisible characters in layout.
- `showsInvisibleCharacters` (page 1513)
  Indicates whether the receiver substitutes visible glyphs for whitespace and other typically invisible characters in layout.
- `setShowsControlCharacters:` (page 1507)
  Specifies whether to substitute visible glyphs for control characters in layout.

- `showsControlCharacters` (page 1513)
  Indicates whether the receiver substitutes visible glyphs for control characters.
- `layoutOptions` (page 1485)
  Returns the layout manager's current layout options.

Controlling Hyphenation

- `setHyphenationFactor:` (page 1503)
  Sets the threshold controlling when hyphenation is done.
- `hyphenationFactor` (page 1478)
  Returns the current hyphenation threshold.

Finding Characters and Glyphs Not Laid Out

- `getFirstUnlaidCharacterIndex:glyphIndex:` (page 1468)
  Returns the indexes for the first character and glyph that have invalid layout information.
- `firstUnlaidCharacterIndex` (page 1467)
  Returns the index for the first character in the layout manager that has not been laid out.
- `firstUnlaidGlyphIndex` (page 1467)
  Returns the index for the first glyph in the layout manager that has not been laid out.

Using Screen Fonts

- `setUsesScreenFonts:` (page 1511)
  Controls using screen fonts to calculate layout and display text.
- `usesScreenFonts` (page 1525)
  Indicates whether the receiver uses screen fonts to calculate layout and display text.
- `substituteFontForFont:` (page 1514)
  Returns a screen font suitable for use in place of the given font, if one is available.

Handling Rulers

- `rulerAccessoryViewForTextView:paragraphStyle:ruler:enabled:` (page 1496)
  Returns the the accessory view that the text system uses for its ruler.
- `rulerMarkersForTextView:paragraphStyle:ruler:` (page 1497)
  Returns an array of text ruler objects for the current selection.

Managing the Responder Chain

- `layoutManagerOwnsFirstResponderInWindow:` (page 1485)
  Indicates whether the first responder in the given window is a text view associated with the receiver.

- `firstTextView` (page 1467)
  Returns the first text view in the receiver's series of text views.
- `textViewForBeginningOfSelection` (page 1522)
  Returns the text view containing the first glyph in the selection.

Drawing

- `drawBackgroundForGlyphRange:atPoint:` (page 1458)
  Draws background marks for the given glyph range, which must lie completely within a single text container.
- `drawGlyphsForGlyphRange:atPoint:` (page 1459)
  Draws the glyphs in the given glyph range, which must lie completely within a single text container.
- `drawUnderlineForGlyphRange:underlineType:baselineOffset:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:` (page 1461)
  Draws underlining for the glyphs in a given range.
- `underlineGlyphRange:underlineType:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:` (page 1523)
  Calculates subranges to be underlined for the glyphs in a given range and draws the underlining as appropriate.
- `showPackedGlyphs:length:glyphRange:atPoint:font:color:printingAdjustment:` (page 1512)
  Draws a range of glyphs.
- `drawStrikethroughForGlyphRange:strikethroughType:baselineOffset:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:` (page 1460)
  Draws a strikethrough for the glyphs in a given range.
- `strikethroughGlyphRange:strikethroughType:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:` (page 1514)
  Calculates and draws strikethrough for the glyphs in the given range.
- `fillBackgroundRectArray:count:forCharacterRange:color:` (page 1466)
  Fills background rectangles with a color.

Accessing the Delegate

- `setDelegate:` (page 1501)
  Sets the receiver's delegate.
- `delegate` (page 1457)
  Returns the receiver's delegate.

Accessing the Typesetter

- `setTypesetter:` (page 1510)
  Sets the current typesetter.
- `typesetter` (page 1522)
  Returns the receiver's typesetter.

Managing Typesetter Compatibility

- `setTypesetterBehavior:` (page 1510)
  Sets the default typesetter behavior.
- `typesetterBehavior` (page 1522)
  Returns the current typesetter behavior.
- `defaultLineHeightForFont:` (page 1456)
  Returns the default line height for a line of text drawn using a given font.
- `defaultBaselineOffsetForFont:` (page 1456)
  Returns the default baseline offset specified by the layout manager's typesetter behavior for the given font.

Managing Temporary Attribute Support

- `addTemporaryAttributes:forCharacterRange:` (page 1449)
  Appends one or more temporary attributes to the attributes dictionary of the specified character range.
- `addTemporaryAttribute:value:forCharacterRange:` (page 1448)
  Adds a temporary attribute with the given name and value to the characters in the specified range.
- `setTemporaryAttributes:forCharacterRange:` (page 1508)
  Sets one or more temporary attributes for the specified character range.
- `removeTemporaryAttribute:forCharacterRange:` (page 1494)
  Removes a temporary attribute from the list of attributes for the specified character range.
- `temporaryAttribute:atCharacterIndex:effectiveRange:` (page 1515)
  Returns the value for the temporary attribute with a given name of the character at a given index, and by reference the range over which the attribute applies.
- `temporaryAttribute:atCharacterIndex:longestEffectiveRange:inRange:` (page 1516)
  Returns the value for the temporary attribute with a given name of the character at a given index, and by reference the maximum range over which the attribute applies.
- `temporaryAttributesAtCharacterIndex:effectiveRange:` (page 1517)
  Returns the dictionary of temporary attributes for the character range specified in *`effectiveCharRange`* at character index *`charIndex`*.
- `temporaryAttributesAtCharacterIndex:longestEffectiveRange:inRange:` (page 1517)
  Returns the temporary attributes for the character at a given index, and by reference the maximum range over which the attributes apply.

Managing Noncontiguous Layout

- `setAllowsNonContiguousLayout:` (page 1498)
  Enables or disables noncontiguous layout.
- `allowsNonContiguousLayout` (page 1450)
  Indicates whether noncontiguous layout is enabled or disabled.
- `hasNonContiguousLayout` (page 1477)
  Indicates whether the layout manager currently has any areas of noncontiguous layout.

- `ensureGlyphsForCharacterRange:` (page 1462)
  Forces the receiver to generate glyphs for the specified character range, if it has not already done so.
- `ensureGlyphsForGlyphRange:` (page 1462)
  Forces the receiver to generate glyphs for the specified glyph range, if it has not already done so.
- `ensureLayoutForCharacterRange:` (page 1463)
  Forces the receiver to perform layout for the specified character range, if it has not already done so.
- `ensureLayoutForGlyphRange:` (page 1463)
  Forces the receiver to perform layout for the specified glyph range, if it has not already done so.
- `ensureLayoutForTextContainer:` (page 1464)
  Forces the receiver to perform layout for the specified text container, if it has not already done so.
- `ensureLayoutForBoundingRect:inTextContainer:` (page 1463)
  Forces the receiver to perform layout for the specified area in the specified text container, if it has not already done so.

Accessing the Font Leading

- `usesFontLeading` (page 1524)
  Indicates whether the receiver uses the leading provided in the font.
- `setUsesFontLeading:` (page 1511)
  Specifies whether or not the receiver uses the leading provided in the font.

Instance Methods

addTemporaryAttribute:value:forCharacterRange:

Adds a temporary attribute with the given name and value to the characters in the specified range.

```
- (void)addTemporaryAttribute:(NSString *)attrName value:(id)value
    forCharacterRange:(NSRange)charRange
```

Parameters

attrName
The name of a temporary attribute.

value
The temporary attribute value associated with *attrName*.

charRange
The range of characters to which the specified attribute-value pair applies.

Discussion
Raises an `NSInvalidArgumentException` if *attrName* or *value* is `nil`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `addTemporaryAttributes:forCharacterRange:` (page 1449)

- setTemporaryAttributes:forCharacterRange: (page 1508)
- removeTemporaryAttribute:forCharacterRange: (page 1494)
- temporaryAttributesAtCharacterIndex:effectiveRange: (page 1517)

Declared In
NSLayoutManager.h

addTemporaryAttributes:forCharacterRange:

Appends one or more temporary attributes to the attributes dictionary of the specified character range.

```
- (void)addTemporaryAttributes:(NSDictionary *)attrs
    forCharacterRange:(NSRange)charRange
```

Parameters

attrs
Attributes dictionary containing the temporary attributes to add.

charRange
The range of characters to which the specified attributes apply.

Discussion
Temporary attributes are used only for onscreen drawing and are not persistent in any way. NSTextView uses them to color misspelled words when continuous spell checking is enabled. Currently the only temporary attributes recognized are those that do not affect layout (colors, underlines, and so on).

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTemporaryAttributes:forCharacterRange: (page 1508)
- removeTemporaryAttribute:forCharacterRange: (page 1494)
- temporaryAttributesAtCharacterIndex:effectiveRange: (page 1517)

Related Sample Code
LayoutManagerDemo

Declared In
NSLayoutManager.h

addTextContainer:

Appends the given text container to the series of text containers where the receiver arranges text.

```
- (void)addTextContainer:(NSTextContainer *)aTextContainer
```

Parameters

aTextContainer
The text container to append.

Discussion
Invalidates glyphs and layout as needed, but doesn't perform glyph generation or layout.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `insertTextContainer:atIndex:` (page 1480)
- `removeTextContainerAtIndex:` (page 1495)
- `textContainers` (page 1520)
- `invalidateGlyphsForCharacterRange:changeInLength:actualCharacterRange:` (page 1482)
- `invalidateLayoutForCharacterRange:isSoft:actualCharacterRange:` (page 1483)

Related Sample Code
QuickLookSketch
Sketch+Accessibility
Sketch-112
TextLayoutDemo
TextViewConfig

Declared In
`NSLayoutManager.h`

allowsNonContiguousLayout

Indicates whether noncontiguous layout is enabled or disabled.

```
- (BOOL)allowsNonContiguousLayout
```

Return Value
`YES` if noncontiguous layout is enabled; otherwise, `NO`.

Discussion
For more information about noncontiguous layout, see "Noncontiguous Layout" (page 1438).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setAllowsNonContiguousLayout:` (page 1498)
- `hasNonContiguousLayout` (page 1477)

Declared In
`NSLayoutManager.h`

attachmentSizeForGlyphAtIndex:

For a glyph corresponding to an attachment, returns the size for the attachment cell to occupy.

```
- (NSSize)attachmentSizeForGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters
glyphIndex
The index of the attachment glyph.

Return Value
The size for the attachment cell to occupy. Returns `{-1.0, -1.0}` if there is no attachment laid for the specified glyph.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAttachmentSize:forGlyphRange:` (page 1498)
- `defaultAttachmentScaling` (page 1456)

Declared In
`NSLayoutManager.h`

attributedString

Returns the text storage object from which the `NSGlyphGenerator` object procures characters for glyph generation.

```
- (NSAttributedString *)attributedString
```

Return Value
The receiver's text storage object.

Discussion
This method is part of the `NSGlyphStorage` protocol, for use by the glyph generator. For `NSLayoutManager` the attributed string is equivalent to the text storage.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSLayoutManager.h`

backgroundLayoutEnabled

Indicates whether the receiver generates glyphs and lays out text when the application's run loop is idle.

```
- (BOOL)backgroundLayoutEnabled
```

Return Value
`YES` if the receiver generates glyphs and lays out text when the application's run loop is idle, `NO` if it performs glyph generation and layout only when necessary.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBackgroundLayoutEnabled:` (page 1499)

Declared In
`NSLayoutManager.h`

boundingRectForGlyphRange:inTextContainer:

Returns a single bounding rectangle (in container coordinates) enclosing all glyphs and other marks drawn in the given text container for the given glyph range, including glyphs that draw outside their line fragment rectangles and text attributes such as underlining.

```
- (NSRect)boundingRectForGlyphRange:(NSRange)glyphRange
    inTextContainer:(NSTextContainer *)container
```

Parameters

glyphRange

The range of glyphs for which to return the bounding rectangle.

container

The text container in which the glyphs are laid out.

Return Value

The bounding rectangle enclosing the given range of glyphs.

Discussion

The range is intersected with the container's range before computing the bounding rectangle. This method can be used to translate glyph ranges into display rectangles for invalidation and redrawing when a range of glyphs changes. Bounding rectangles are always in container coordinates.

Performs glyph generation and layout if needed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `glyphRangeForBoundingRect:inTextContainer:` (page 1475)
- `glyphRangeForTextContainer:` (page 1477)
- `drawsOutsideLineFragmentForGlyphAtIndex:` (page 1459)

Related Sample Code

LayoutManagerDemo

Declared In

`NSLayoutManager.h`

boundsRectForTextBlock:atIndex:effectiveRange:

Returns the bounding rectangle within which the given text block containing the glyph at the given index is to be laid out.

```
- (NSRect)boundsRectForTextBlock:(NSTextBlock *)block atIndex:(NSUInteger)glyphIndex
    effectiveRange:(NSRangePointer)effectiveGlyphRange
```

Parameters

block

The text block whose bounding rectangle is returned.

glyphIndex

Index of the glyph.

effectiveGlyphRange
If not NULL, on output, the range for all glyphs in the text block.

Return Value
The bounding rectangle of the text block, or NSZeroRect if no rectangle has been set for the specified block since the last invalidation.

Discussion
This method causes glyph generation but not layout. Block layout rectangles and bounds rectangles are always in container coordinates.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setBoundsRect:forTextBlock:glyphRange: (page 1499)

Declared In
NSLayoutManager.h

boundsRectForTextBlock:glyphRange:

Returns the bounding rectangle enclosing the given text block containing the given glyph range.

```
- (NSRect)boundsRectForTextBlock:(NSTextBlock *)block glyphRange:(NSRange)glyphRange
```

Parameters
block
The text block whose bounds rectangle is returned.

glyphRange
The range of glyphs in the text block.

Return Value
The bounding rectangle, or NSZeroRect if no rectangle has been set for the specified block since the last invalidation

Discussion
This method causes glyph generation but not layout. Block layout rectangles and bounds rectangles are always in container coordinates.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setBoundsRect:forTextBlock:glyphRange: (page 1499)

Declared In
NSLayoutManager.h

characterIndexForGlyphAtIndex:

Returns the index in the text storage for the first character associated with the given glyph.

```
- (NSUInteger)characterIndexForGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

glyphIndex

The index of the glyph for which to return the associated character.

Return Value

The index of the first character associated with the glyph at the specified index.

Discussion

If noncontiguous layout is not enabled, this method causes generation of all glyphs up to and including *glyphIndex*. This method accepts an index beyond the last glyph, returning an index extrapolated from the last actual glyph index.

In many cases it's better to use the range-mapping methods, `characterRangeForGlyphRange:actualGlyphRange:` (page 1455) and `glyphRangeForCharacterRange:actualCharacterRange:` (page 1476), which provide more comprehensive information.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- glyphIndexForCharacterAtIndex:` (page 1473)

Related Sample Code

LayoutManagerDemo

Declared In

`NSLayoutManager.h`

characterIndexForPoint:inTextContainer:fractionOfDistanceBetweenInsertionPoints:

Returns the index of the character falling under the given point, expressed in the given container's coordinate system.

```
- (NSUInteger)characterIndexForPoint:(NSPoint)point inTextContainer:(NSTextContainer
    *)container fractionOfDistanceBetweenInsertionPoints:(CGFloat *)partialFraction
```

Parameters

point

The point to test.

container

The text container within which the point is tested.

partialFraction

A fraction of the distance from the insertion point, logically before the given character to the next one.

Return Value

The index of the character falling under *point*.

Discussion

Analogous to `glyphIndexForPoint:inTextContainer:fractionOfDistanceThroughGlyph:` (page 1474), but expressed in character index terms. The method returns the index of the character falling under *point*, expressed in coordinate system of *container*; if no character is under the point, the nearest character is returned, where nearest is defined according to the requirements of selection by mouse. However, this is

not simply equivalent to taking the result of the corresponding glyph index method and converting it to a character index, because in some cases a single glyph represents more than one selectable character, for example an fi ligature glyph. In that case, there is an insertion point within the glyph, and this method returns one character or the other, depending on whether the specified point lies to the left or the right of that insertion point.

In general, this method returns only character indexes for which there is an insertion point. The *partialFraction* is a fraction of the distance from the insertion point, logically before the given character to the next one, which may be either to the right or to the left depending on directionality.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSLayoutManager.h

characterRangeForGlyphRange:actualGlyphRange:

Returns the range of characters that generated the glyphs in the given glyph range.

```
- (NSRange)characterRangeForGlyphRange:(NSRange)glyphRange
    actualGlyphRange:(NSRangePointer)actualGlyphRange
```

Parameters

glyphRange

The glyph range for which to return the character range.

actualGlyphRange

If not NULL, on output, points to the full range of glyphs generated by the character range returned. This range may be identical or slightly larger than the requested glyph range. For example, if the text storage contains the character "Ö" and the glyph cache contains the two atomic glyphs "O" and "¨", and if *glyphRange* encloses only the first or second glyph, then *actualGlyphRange* is set to enclose both glyphs.

Return Value
The range of characters that generated the glyphs in *glyphRange*.

Discussion
If the length of *glyphRange* is 0, the resulting character range is a zero-length range just after the character(s) corresponding to the preceding glyph, and *actualGlyphRange* is also zero-length. If *glyphRange* extends beyond the text length, the method truncates the result to the number of characters in the text.

If noncontiguous layout is not enabled, this method forces the generation of glyphs for all characters up to and including the end of the returned range.

Availability
Available in Mac OS X v10.0 and later.

See Also
- characterIndexForGlyphAtIndex: (page 1453)
- glyphRangeForCharacterRange:actualCharacterRange: (page 1476)

Related Sample Code
LayoutManagerDemo
TipWrapper

Declared In
NSLayoutManager.h

defaultAttachmentScaling

Returns the default behavior desired if an attachment image is too large to fit in a text container.

```
- (NSImageScaling)defaultAttachmentScaling
```

Discussion
Attachment cells control their own size and drawing, so this setting is only advisory to them, but Application Kit–supplied attachment cells respect it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setDefaultAttachmentScaling: (page 1500)

Declared In
NSLayoutManager.h

defaultBaselineOffsetForFont:

Returns the default baseline offset specified by the layout manager's typesetter behavior for the given font.

```
- (CGFloat)defaultBaselineOffsetForFont:(NSFont *)theFont
```

Parameters
theFont
The font for which to return the default baseline offset.

Return Value
The default baseline offset for a line of text drawn using *theFont*.

Discussion
The value returned may vary according to the layout manager's typesetter behavior.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setTypesetterBehavior: (page 1510)
- defaultLineHeightForFont: (page 1456)

Declared In
NSLayoutManager.h

defaultLineHeightForFont:

Returns the default line height for a line of text drawn using a given font.

```
- (CGFloat)defaultLineHeightForFont:(NSFont *)theFont
```

Parameters

theFont

The font for which to determine the default line height.

Return Value

The default line height for a line of text drawn using *theFont*.

Discussion

The value returned may vary according to the layout manager's typesetter behavior.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `setTypesetterBehavior:` (page 1510)
- `defaultBaselineOffsetForFont:` (page 1456)

Declared In

`NSLayoutManager.h`

delegate

Returns the receiver's delegate.

```
- (id < NSLayoutManagerDelegate >)delegate
```

Return Value

The receiver's delegate.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setDelegate:` (page 1501)

Declared In

`NSLayoutManager.h`

deleteGlyphsInRange:

Deletes the glyphs in the given range from the receiver's glyph store.

```
- (void)deleteGlyphsInRange:(NSRange)glyphRange
```

Parameters

glyphRange

The range of glyphs to delete.

Discussion
This method is for use by the glyph-generation mechanism and doesn't perform any invalidation or generation of the glyphs or layout. This method should be invoked only during glyph generation and typesetting, in almost all cases only by the glyph generator or typesetter. For example, a custom glyph generator or typesetter might invoke it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `insertGlyph:atGlyphIndex:characterIndex:` (page 1479)

Declared In
`NSLayoutManager.h`

drawBackgroundForGlyphRange:atPoint:

Draws background marks for the given glyph range, which must lie completely within a single text container.

```
- (void)drawBackgroundForGlyphRange:(NSRange)glyphsToShow atPoint:(NSPoint)origin
```

Parameters

glyphsToShow
The range of glyphs for which the background is drawn.

origin
The position of the text container in the coordinate system of the currently focused view.

Discussion
This method is called by `NSTextView` for drawing. You can override it to perform additional drawing, or to replace text drawing entirely, but not to change layout. You can call this method directly, but focus must already be locked on the destination view or image.

Background marks are such things as selection highlighting, text background color, and any background for marked text, along with block decoration such as table backgrounds and borders.

Performs glyph generation and layout if needed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawGlyphsForGlyphRange:atPoint:` (page 1459)
- `glyphRangeForTextContainer:` (page 1477)
- `fillBackgroundRectArray:count:forCharacterRange:color:` (page 1466)
- `textContainerOrigin` (page 2956) (`NSTextView`)

Related Sample Code
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSLayoutManager.h`

drawGlyphsForGlyphRange:atPoint:

Draws the glyphs in the given glyph range, which must lie completely within a single text container.

```
- (void)drawGlyphsForGlyphRange:(NSRange)glyphsToShow atPoint:(NSPoint)origin
```

Parameters

glyphsToShow

The range of glyphs that are drawn.

origin

The position of the text container in the coordinate system of the currently focused view.

Discussion

This method is called by `NSTextView` for drawing. You can override it to perform additional drawing, or to replace text drawing entirely, but not to change layout. You can call this method directly, but focus must already be locked on the destination view or image. This method expects the coordinate system of the view to be flipped.

This method draws the actual glyphs, including attachments, as well as any underlines or strikethoughs.

Performs glyph generation and layout if needed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `drawBackgroundForGlyphRange:atPoint:` (page 1458)
- `glyphRangeForTextContainer:` (page 1477)
- `textContainerOrigin` (page 2956) (`NSTextView`)

Related Sample Code

CircleView
DockTile
QuickLookSketch
Sketch-112
SpeedometerView

Declared In

`NSLayoutManager.h`

drawsOutsideLineFragmentForGlyphAtIndex:

Indicates whether the glyph draws outside of its line fragment rectangle.

```
- (BOOL)drawsOutsideLineFragmentForGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

glyphIndex

Index of the glyph.

Return Value

`YES` if the glyph at *glyphIndex* exceeds the bounds of the line fragment where it's laid out, `NO` otherwise.

Discussion
Exceeding bounds can happen when text is set at a fixed line height. For example, if the user specifies a fixed line height of 12 points and sets the font size to 24 points, the glyphs will exceed their layout rectangles.

This method causes glyph generation and layout for the line fragment containing the specified glyph, or if noncontiguous layout is not enabled, up to and including that line fragment.

Glyphs that draw outside their line fragment rectangles aren't considered when calculating enclosing rectangles with the `rectArrayForCharacterRange:withinSelectedCharacterRange:inTextContainer:rectCount:` (page 1492) and `rectArrayForGlyphRange:withinSelectedGlyphRange:inTextContainer:rectCount:` (page 1493) methods. They are, however, considered by `boundingRectForGlyphRange:inTextContainer:` (page 1452).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSLayoutManager.h`

drawStrikethroughForGlyphRange:strikethroughType:baselineOffset:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:

Draws a strikethrough for the glyphs in a given range.

```
- (void)drawStrikethroughForGlyphRange:(NSRange)glyphRange
    strikethroughType:(NSInteger)strikethroughVal
    baselineOffset:(CGFloat)baselineOffset lineFragmentRect:(NSRect)lineRect
    lineFragmentGlyphRange:(NSRange)lineGlyphRange
    containerOrigin:(NSPoint)containerOrigin
```

Parameters

glyphRange
The range of glyphs for which to draw a strikethrough. The range must belong to a single line fragment rectangle (as returned by `lineFragmentRectForGlyphAtIndex:effectiveRange:` (page 1487)).

strikethroughVal
The style of strikethrough to draw. This value is a mask derived from the value for `NSUnderlineStyleAttributeName` (page 272)—for example, `(NSUnderlinePatternDash | NSUnderlineStyleThick)`. Subclasses can define custom strikethrough styles.

baselineOffset
Indicates how far above the text baseline the underline should be drawn.

lineRect
The line fragment rectangle containing the glyphs to draw strikethrough for.

lineGlyphRange
The range of all glyphs within *lineRect*.

containerOrigin
The origin of the line fragment rectangle's `NSTextContainer` in its `NSTextView`.

Discussion
This method is invoked automatically by `strikethroughGlyphRange:strikethroughType:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:` (page 1514); you should rarely need to invoke it directly. This method's *strikethroughVal* parameter does not take account of any setting for`NSUnderlineByWordMask` (page 276) because that's taken care of by `underlineGlyphRange:underlineType:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:` (page 1523).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSLayoutManager.h`

drawUnderlineForGlyphRange:underlineType:baselineOffset:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:

Draws underlining for the glyphs in a given range.

```
- (void)drawUnderlineForGlyphRange:(NSRange)glyphRange
    underlineType:(NSInteger)underlineVal baselineOffset:(CGFloat)baselineOffset
    lineFragmentRect:(NSRect)lineRect lineFragmentGlyphRange:(NSRange)lineGlyphRange
     containerOrigin:(NSPoint)containerOrigin
```

Parameters

glyphRange
A range of glyphs, which must belong to a single line fragment rectangle (as returned by `lineFragmentRectForGlyphAtIndex:effectiveRange:` (page 1487)).

underlineVal
The style of underlining to draw. This value is a mask derived from the value for `NSUnderlineStyleAttributeName` (page 272)—for example, `(NSUnderlinePatternDash | NSUnderlineStyleThick)`. Subclasses can define custom underlining styles.

baselineOffset
Specifies the distance from the bottom of the bounding box of the specified glyphs in the specified range to their baseline.

lineRect
The line fragment rectangle containing the glyphs to draw underlining for.

lineGlyphRange
The range of all glyphs within *lineRect*.

containerOrigin
The origin of the *lineRect*`NSTextContainer` in its `NSTextView`.

Discussion
This method is invoked automatically by `underlineGlyphRange:underlineType:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:` (page 1523); you should rarely need to invoke it directly. This method's *underlineVal* parameter does not take account of any setting for`NSUnderlineByWordMask` (page 276) because that's taken care of by `underlineGlyphRange:underlineType:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:` (page 1523).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textContainerForGlyphAtIndex:effectiveRange:` (page 1519)
- `textContainerOrigin` (page 2956) (`NSTextView`)

Declared In
`NSLayoutManager.h`

ensureGlyphsForCharacterRange:

Forces the receiver to generate glyphs for the specified character range, if it has not already done so.

```
- (void)ensureGlyphsForCharacterRange:(NSRange)charRange
```

Parameters

charRange
The character range for which glyphs are generated.

Discussion
The layout manager reserves the right to perform glyph generation for larger ranges. If noncontiguous layout is disabled, then the affected range is always effectively extended to start at the beginning of the text.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSLayoutManager.h`

ensureGlyphsForGlyphRange:

Forces the receiver to generate glyphs for the specified glyph range, if it has not already done so.

```
- (void)ensureGlyphsForGlyphRange:(NSRange)glyphRange
```

Parameters

glyphRange
The glyph range for which glyphs are generated.

Discussion
The layout manager reserves the right to perform glyph generation for larger ranges. If noncontiguous layout is disabled, then the affected range is always effectively extended to start at the beginning of the text.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSLayoutManager.h`

ensureLayoutForBoundingRect:inTextContainer:

Forces the receiver to perform layout for the specified area in the specified text container, if it has not already done so.

```
- (void)ensureLayoutForBoundingRect:(NSRect)bounds inTextContainer:(NSTextContainer
    *)container
```

Parameters

bounds

The area for which layout is performed.

container

The text container containing the area for which layout is performed.

Discussion

The layout manager reserves the right to perform layout for larger ranges. If noncontiguous layout is disabled, then the affected range is always effectively extended to start at the beginning of the text.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSLayoutManager.h`

ensureLayoutForCharacterRange:

Forces the receiver to perform layout for the specified character range, if it has not already done so.

```
- (void)ensureLayoutForCharacterRange:(NSRange)charRange
```

Parameters

charRange

The character range for which layout is performed.

Discussion

The layout manager reserves the right to perform layout for larger ranges. If noncontiguous layout is disabled, then the affected range is always effectively extended to start at the beginning of the text.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSLayoutManager.h`

ensureLayoutForGlyphRange:

Forces the receiver to perform layout for the specified glyph range, if it has not already done so.

```
- (void)ensureLayoutForGlyphRange:(NSRange)glyphRange
```

Parameters

glyphRange

The glyph range for which layout is performed.

Discussion
The layout manager reserves the right to perform layout for larger ranges. If noncontiguous layout is disabled, then the affected range is always effectively extended to start at the beginning of the text.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSLayoutManager.h`

ensureLayoutForTextContainer:

Forces the receiver to perform layout for the specified text container, if it has not already done so.

```
- (void)ensureLayoutForTextContainer:(NSTextContainer *)container
```

Parameters

container
The text container for which layout is performed.

Discussion
The layout manager reserves the right to perform layout for larger ranges. If noncontiguous layout is disabled, then the affected range is always effectively extended to start at the beginning of the text.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSLayoutManager.h`

extraLineFragmentRect

Returns the rectangle defining the extra line fragment for the insertion point at the end of a text (either in an empty text or after a final paragraph separator).

```
- (NSRect)extraLineFragmentRect
```

Return Value
The rectangle defining the extra line fragment for the insertion point.

Discussion
The rectangle is defined in the coordinate system of its `NSTextContainer`. Returns `NSZeroRect` if there is no such rectangle.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `extraLineFragmentUsedRect` (page 1465)
- `extraLineFragmentTextContainer` (page 1465)
- `setExtraLineFragmentRect:usedRect:textContainer:` (page 1502)

Declared In
NSLayoutManager.h

extraLineFragmentTextContainer

Returns the text container that contains the extra line fragment rectangle.

```
- (NSTextContainer *)extraLineFragmentTextContainer
```

Return Value
The text container that contains the extra line fragment rectangle, or `nil` if there is no extra line fragment rectangle.

Discussion
This rectangle is used to display the insertion point at the end of a text (either in an empty text or after a final paragraph separator).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `extraLineFragmentRect` (page 1464)
- `extraLineFragmentUsedRect` (page 1465)
- `setExtraLineFragmentRect:usedRect:textContainer:` (page 1502)

Declared In
NSLayoutManager.h

extraLineFragmentUsedRect

Returns the rectangle enclosing the insertion point drawn in the extra line fragment rectangle.

```
- (NSRect)extraLineFragmentUsedRect
```

Return Value
The rectangle enclosing the insertion point.

Discussion
The rectangle is defined in the coordinate system of its `NSTextContainer`. Returns `NSZeroRect` if there is no extra line fragment rectangle.

The extra line fragment used rectangle is twice as wide (or tall) as the text container's line fragment padding, with the insertion point itself in the middle.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `extraLineFragmentRect` (page 1464)
- `extraLineFragmentTextContainer` (page 1465)
- `setExtraLineFragmentRect:usedRect:textContainer:` (page 1502)

Declared In
NSLayoutManager.h

fillBackgroundRectArray:count:forCharacterRange:color:

Fills background rectangles with a color.

```
- (void)fillBackgroundRectArray:(NSRectArray)rectArray count:(NSUInteger)rectCount
    forCharacterRange:(NSRange)charRange color:(NSColor *)color
```

Parameters

rectArray
The array of rectangles to fill.

rectCount
The number of rectangles in *rectArray*.

charRange
The range of characters whose background rectangles are filled.

color
The fill color.

Discussion
This is the primitive method used by drawBackgroundForGlyphRange:atPoint: (page 1458), providing a finer-grained override point for actually filling rectangles with a particular background color for a background color attribute, a selected or marked range highlight, a block decoration, or any other rectangle fill needed by that method. As with showPackedGlyphs:length:glyphRange:atPoint:font:color:printingAdjustment: (page 1512), the *charRange* and *color* parameters are passed in merely for informational purposes; the color is already set in the graphics state. If for any reason you modify it, you must restore it before returning from this method.

This method operates in terms of character ranges, because the relevant attributes are expressed on characters, and they don't always lie on glyph boundaries—for example, when one character of an "fi" ligature is highlighted.

You should never call this method, but you might override it. The default implementation simply fills the rectangles in the specified array. The graphics operation used for this fill is controlled by a link check; for compatibility reasons, it is NSCompositeCopy (page 1376) for applications linked prior to Mac OS X v10.6 and NSCompositeSourceOver (page 1376) for applications linked on Mac OS X v10.6 or later. Applications can control the operation used, or modify the drawing, by overriding this method in an NSLayoutManager subclass.

Availability
Available in Mac OS X v10.6 and later.

See Also
- drawBackgroundForGlyphRange:atPoint: (page 1458)

Declared In
NSLayoutManager.h

firstTextView

Returns the first text view in the receiver's series of text views.

```
- (NSTextView *)firstTextView
```

Return Value
The receiver's first text view.

Discussion
This `NSTextView` object is the recipient of various `NSText` and `NSTextView` notifications.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
QuickLookSketch
Sketch+Accessibility

Declared In
`NSLayoutManager.h`

firstUnlaidCharacterIndex

Returns the index for the first character in the layout manager that has not been laid out.

```
- (NSUInteger)firstUnlaidCharacterIndex
```

Return Value
The character index.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSLayoutManager.h`

firstUnlaidGlyphIndex

Returns the index for the first glyph in the layout manager that has not been laid out.

```
- (NSUInteger)firstUnlaidGlyphIndex
```

Return Value
The glyph index.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSLayoutManager.h`

fractionOfDistanceThroughGlyphForPoint:inTextContainer:

This method is a primitive for `glyphIndexForPoint:inTextContainer:fractionOfDistanceThroughGlyph:` (page 1474). You should always call the main method, not the primitives.

```
- (CGFloat)fractionOfDistanceThroughGlyphForPoint:(NSPoint)point
    inTextContainer:(NSTextContainer *)container
```

Discussion
Overriding should be done for the primitive methods. Existing subclasses that do not do this overriding will not have their implementations available to Java developers.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- glyphIndexForPoint:inTextContainer:` (page 1473)

Declared In
`NSLayoutManager.h`

getFirstUnlaidCharacterIndex:glyphIndex:

Returns the indexes for the first character and glyph that have invalid layout information.

```
- (void)getFirstUnlaidCharacterIndex:(NSUInteger *)charIndex glyphIndex:(NSUInteger
    *)glyphIndex
```

Parameters

charIndex
On return, if not `NULL`, the index of the first character that has invalid layout information

glyphIndex
On return, if not `NULL`, the index of the first glyph that has invalid layout information.

Discussion
Either parameter may be `NULL`, in which case the receiver simply ignores it.

As part of its implementation, this method calls `firstUnlaidCharacterIndex` (page 1467) and `firstUnlaidGlyphIndex` (page 1467). To change this method's behavior, override those two methods instead of this one.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSLayoutManager.h`

getGlyphs:range:

Fills the passed-in buffer with a sequence of glyphs

```
- (NSUInteger)getGlyphs:(NSGlyph *)glyphArray range:(NSRange)glyphRange
```

Parameters

glyphArray

On output, the displayable glyphs from *glyphRange*, null-terminated. Does not include in the result any NSNullGlyph or other glyphs that are not shown. The memory passed in should be large enough for at least glyphRange.length+1 elements.

glyphRange

The range of glyphs from which to return the displayable glyphs.

Return Value

The actual number of glyphs filled into the array is returned (not counting the null-termination).

Discussion

Raises an NSRangeException if the range specified exceeds the bounds of the actual glyph range for the receiver. Performs glyph generation if needed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- glyphAtIndex: (page 1471)
- glyphAtIndex:isValidIndex: (page 1472)
- notShownAttributeForGlyphAtIndex: (page 1490)

Declared In

NSLayoutManager.h

getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:

Returns the glyphs and information needed to perform layout for the given glyph range.

```
- (NSUInteger)getGlyphsInRange:(NSRange)glyphRange glyphs:(NSGlyph *)glyphBuffer
    characterIndexes:(NSUInteger *)charIndexBuffer
    glyphInscriptions:(NSGlyphInscription *)inscribeBuffer elasticBits:(BOOL
    *)elasticBuffer
```

Discussion

This is a convenience method for getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:bidiLevels: (page 1469) that does not return a *bidiLevelBuffer*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSLayoutManager.h

getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:bidiLevels:

Returns the glyphs and information needed to perform layout for the given glyph range.

```
- (NSUInteger)getGlyphsInRange:(NSRange)glyphRange glyphs:(NSGlyph *)glyphBuffer
    characterIndexes:(NSUInteger *)charIndexBuffer
    glyphInscriptions:(NSGlyphInscription *)inscribeBuffer elasticBits:(BOOL
    *)elasticBuffer bidiLevels:(unsigned char *)bidiLevelBuffer
```

Parameters

glyphRange

The range of glyphs to lay out.

glyphBuffer

On output, the sequence of glyphs needed to lay out the given glyph range.

charIndexBuffer

On output, the indexes of the original characters corresponding to the given glyph range. Note that a glyph at index 1 is not necessarily mapped to the character at index 1, since a glyph may be for a ligature or accent.

inscribeBuffer

On output, the inscription attributes for each glyph, which are used to lay out characters that are combined together. The possible values are described in "Constants" (page 1525).

elasticBuffer

On output, values indicating whether a glyph is elastic for each glyph. An elastic glyph can be made longer at the end of a line or when needed for justification.

bidiLevelBuffer

On output, the direction of each glyph for bidirectional text. The values range from 0 to 61 as defined by Unicode Standard Annex #9. An even value means the glyph goes left-to-right, and an odd value means the glyph goes right-to-left.

Return Value

The number of glyphs returned in *glyphBuffer*.

Discussion

This method and `getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:` (page 1469) are intended primarily to enable the typesetter to obtain in bulk the glyphs and other information that it needs to perform layout. These methods return all glyphs in the range, including `NSNullGlyph` and not-shown glyphs. They do not null-terminate the results. Each pointer passed in should either be `NULL`, or else point to sufficient memory to hold `glyphRange.length` elements.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSLayoutManager.h`

getLineFragmentInsertionPointsForCharacterAtIndex:alternatePositions: inDisplayOrder:positions:characterIndexes:

Returns insertion points in bulk for a given line fragment.

```
- (NSUInteger)getLineFragmentInsertionPointsForCharacterAtIndex:(NSUInteger)charIndex
    alternatePositions:(BOOL)aFlag inDisplayOrder:(BOOL)dFlag positions:(CGFloat
    *)positions characterIndexes:(NSUInteger *)charIndexes
```

Parameters

charIndex

The character index of one character within the line fragment.

aFlag

If YES, returns alternate, rather than primary, insertion points.

dFlag

If YES, returns insertion points in display, rather than logical, order.

positions

On output, the positions of the insertion points, in the order specified.

charIndexes

On output, the indexes of the characters corresponding to the returned insertion points.

Return Value
The number of insertion points returned.

Discussion
The method allows clients to obtain all insertion points for a line fragment in one call. Each pointer passed in should either be NULL or else point to sufficient memory to hold as many elements as there are insertion points in the line fragment (which cannot be more than the number of characters + 1). The returned positions indicate a transverse offset relative to the line fragment rectangle's origin. Internal caching is used to ensure that repeated calls to this method for the same line fragment (possibly with differing values for other arguments) are not significantly more expensive than a single call.

Availability
Available in Mac OS X v10.5 and later.

See Also
- rectArrayForCharacterRange:withinSelectedCharacterRange:inTextContainer:rectCount: (page 1492)
- rectArrayForGlyphRange:withinSelectedGlyphRange:inTextContainer:rectCount: (page 1493)

Declared In
NSLayoutManager.h

glyphAtIndex:

Returns the glyph at *glyphIndex*.

```
- (NSGlyph)glyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

glyphIndex

The index of a glyph in the receiver. This value must not exceed the bounds of the receiver's glyph array.

Return Value
The glyph at *glyphIndex*.

Discussion
Raises an NSRangeException if *glyphIndex* is out of bounds.

Performs glyph generation if needed. To avoid an exception with `glyphAtIndex:` you must first check the glyph index against the number of glyphs, which requires generating all glyphs. Another method, `glyphAtIndex:isValidIndex:` (page 1472), generates glyphs only up to the one requested, so using it can be more efficient.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- getGlyphs:range:` (page 1468)

Declared In
`NSLayoutManager.h`

glyphAtIndex:isValidIndex:

If the given index is valid, returns the glyph at that location and optionally returns a flag indicating whether the requested index is in range.

```
- (NSGlyph)glyphAtIndex:(NSUInteger)glyphIndex isValidIndex:(BOOL *)isValidIndex
```

Parameters

glyphIndex
The index of the glyph to be returned.

isValidIndex
If not `NULL`, on output, `YES` if the requested index is in range; otherwise `NO`.

Return Value
The glyph at the requested index, or `NSNullGlyph` if the requested index is out of the range `{0, numberOfGlyphs` (page 1491)`}`.

Discussion
If noncontiguous layout is not enabled, this method causes generation of all glyphs up to and including *glyphIndex*.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- getGlyphs:range:` (page 1468)
`- glyphAtIndex:` (page 1471)

Declared In
`NSLayoutManager.h`

glyphGenerator

Returns the glyph generator used by this layout manager.

```
- (NSGlyphGenerator *)glyphGenerator
```

Return Value
The glyph generator.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setGlyphGenerator:` (page 1502)

Declared In
`NSLayoutManager.h`

glyphIndexForCharacterAtIndex:

Returns the index of the first glyph associated with the character at the specified index.

```
- (NSUInteger)glyphIndexForCharacterAtIndex:(NSUInteger)charIndex
```

Parameters

charIndex
The index of the character for which to return the associated glyph.

Return Value
The index of the first glyph associated with the character at the specified index.

Discussion
If noncontiguous layout is not enabled, this method causes generation of all glyphs up to and including those associated with the specified character. This method accepts an index beyond the last character, returning an index extrapolated from the last actual character index.

In many cases it's better to use the range-mapping methods, `characterRangeForGlyphRange:actualGlyphRange:` (page 1455) and `glyphRangeForCharacterRange:actualCharacterRange:` (page 1476), which provide more comprehensive information.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `characterIndexForGlyphAtIndex:` (page 1453)

Declared In
`NSLayoutManager.h`

glyphIndexForPoint:inTextContainer:

This method is a primitive for `glyphIndexForPoint:inTextContainer:fractionOfDistanceThroughGlyph:` (page 1474). You should always call the main method, not the primitives.

```
- (NSUInteger)glyphIndexForPoint:(NSPoint)point inTextContainer:(NSTextContainer
    *)container
```

Discussion
Overriding should be done for the primitive methods. Existing subclasses that do not do this overriding will not have their implementations available to Java developers.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `fractionOfDistanceThroughGlyphForPoint:inTextContainer:` (page 1468)

Related Sample Code
LayoutManagerDemo

Declared In
`NSLayoutManager.h`

glyphIndexForPoint:inTextContainer:fractionOfDistanceThroughGlyph:

Returns the index of the glyph falling under the given point, expressed in the given container's coordinate system.

```
- (NSUInteger)glyphIndexForPoint:(NSPoint)point inTextContainer:(NSTextContainer
    *)container fractionOfDistanceThroughGlyph:(CGFloat *)partialFraction
```

Parameters

point
The point for which to return the glyph, in coordinates of *container*.

container
The container in which the returned glyph is laid out.

partialFraction
If not `NULL`, on output, the fraction of the distance between the location of the glyph returned and the location of the next glyph.

Return Value
The index of the glyph falling under the given point, expressed in the given container's coordinate system.

Discussion
If no glyph is under *point*, the nearest glyph is returned, where nearest is defined according to the requirements of selection by mouse. Clients who wish to determine whether the the point actually lies within the bounds of the glyph returned should follow this with a call to `boundingRectForGlyphRange:inTextContainer:` (page 1452) and test whether the point falls in the rectangle returned by that method. If *partialFraction* is non-NULL, it returns by reference the fraction of the distance between the location of the glyph returned and the location of the next glyph.

For purposes such as dragging out a selection or placing the insertion point, a partial percentage less than or equal to 0.5 indicates that *point* should be considered as falling before the glyph index returned; a partial percentage greater than 0.5 indicates that it should be considered as falling after the glyph index returned. If the nearest glyph doesn't lie under *point* at all (for example, if *point* is beyond the beginning or end of a line), this ratio is 0 or 1.

If the glyph stream contains the glyphs "A" and "b", with the width of "A" being 13 points, and the user clicks at a location 8 points into "A", *partialFraction* is 8/13, or 0.615. In this case, the point given should be considered as falling between "A" and "b" for purposes such as dragging out a selection or placing the insertion point.

Performs glyph generation and layout if needed.

As part of its implementation, this method calls `fractionOfDistanceThroughGlyphForPoint:inTextContainer:` (page 1468) and `glyphIndexForPoint:inTextContainer:` (page 1473). To change this method's behavior, override those two methods instead of this one.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSLayoutManager.h`

glyphRangeForBoundingRect:inTextContainer:

Returns the smallest contiguous range for glyphs that are laid out wholly or partially within the given rectangle in the given text container.

```
- (NSRange)glyphRangeForBoundingRect:(NSRect)bounds inTextContainer:(NSTextContainer
    *)container
```

Parameters

bounds
The bounding rectangle for which to return glyphs.

container
The text container in which the glyphs are laid out.

Return Value
The range of glyphs that would need to be displayed in order to draw all glyphs that fall (even partially) within the given bounding rectangle. The range returned can include glyphs that don't fall inside or intersect *bounds*, although the first and last glyphs in the range always do. At most this method returns the glyph range for the whole container.

Discussion
This method is used to determine which glyphs need to be displayed within a given rectangle.

Performs glyph generation and layout if needed. Bounding rectangles are always in container coordinates.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- glyphRangeForBoundingRectWithoutAdditionalLayout:inTextContainer:` (page 1475)

Declared In
`NSLayoutManager.h`

glyphRangeForBoundingRectWithoutAdditionalLayout:inTextContainer:

Returns the smallest contiguous range for glyphs that are laid out wholly or partially within the given rectangle in the given text container.

```
- (NSRange)glyphRangeForBoundingRectWithoutAdditionalLayout:(NSRect)bounds
    inTextContainer:(NSTextContainer *)container
```

Parameters

bounds

The bounding rectangle for which to return glyphs.

container

The text container in which the glyphs are laid out.

Return Value

The range of glyphs that would need to be displayed in order to draw all glyphs that fall (even partially) within the given bounding rectangle. The range returned can include glyphs that don't fall inside or intersect *bounds*, although the first and last glyphs in the range always do. At most this method returns the glyph range for the whole container.

Discussion

Unlike `glyphRangeForBoundingRect:inTextContainer:` (page 1475), this variant of the method doesn't perform glyph generation or layout. Its results, though faster, can be incorrect. This method is primarily for use by `NSTextView`; you should rarely need to use it yourself.

Bounding rectangles are always in container coordinates.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- glyphRangeForBoundingRect:inTextContainer:` (page 1475)

Declared In

`NSLayoutManager.h`

glyphRangeForCharacterRange:actualCharacterRange:

Returns the range of glyphs that are generated from the characters in the given character range.

```
- (NSRange)glyphRangeForCharacterRange:(NSRange)charRange
    actualCharacterRange:(NSRangePointer)actualCharRange
```

Parameters

charRange

The character range for which to return the generated glyph range.

actualCharRange

If not `NULL`, on output, points to the actual range of characters that fully define the glyph range returned. This range may be identical to or slightly larger than the requested character range. For example, if the text storage contains the characters "O" and "¨", and the glyph store contains the single precomposed glyph "Ö", and if *charRange* encloses only the first or second character, then *actualCharRange* is set to enclose both characters.

Return Value

The range of glyphs generated by *charRange*.

Discussion

If the length of *charRange* is 0, the resulting glyph range is a zero-length range just after the glyph(s) corresponding to the preceding character, and *actualCharRange* will also be zero-length. If *charRange* extends beyond the text length, the method truncates the result to the number of glyphs in the text.

If noncontiguous layout is not enabled, this method forces the generation of glyphs for all characters up to and including the end of the specified range.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `characterIndexForGlyphAtIndex:` (page 1453)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TipWrapper

Declared In
`NSLayoutManager.h`

glyphRangeForTextContainer:

Returns the range of glyphs laid out within the given text container.

```
- (NSRange)glyphRangeForTextContainer:(NSTextContainer *)aTextContainer
```

Discussion
This is a less efficient method than the similar `textContainerForGlyphAtIndex:effectiveRange:` (page 1519).

Performs glyph generation and layout if needed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textContainerForGlyphAtIndex:effectiveRange:` (page 1519)

Related Sample Code
CircleView
Sketch+Accessibility
Sketch-112
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSLayoutManager.h`

hasNonContiguousLayout

Indicates whether the layout manager currently has any areas of noncontiguous layout.

```
- (BOOL)hasNonContiguousLayout
```

Return Value
`YES` if noncontiguous layout exists; otherwise, `NO`.

Discussion
There may be times at which there is no noncontiguous layout, such as when layout is complete; this method enables the layout manager to report that to clients.

For more information about noncontiguous layout, see "Noncontiguous Layout" (page 1438).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `allowsNonContiguousLayout` (page 1450)
- `setAllowsNonContiguousLayout:` (page 1498)

Declared In
`NSLayoutManager.h`

hyphenationFactor

Returns the current hyphenation threshold.

```
- (float)hyphenationFactor
```

Return Value
The hyphenation factor ranging from 0.0 to 1.0. By default, the value is 0.0, meaning hyphenation is off. A value of 1.0 causes hyphenation to be attempted always.

Discussion
Whenever (width of the real contents of the line) / (the line fragment width) is less than `hyphenationFactor`, hyphenation is attempted when laying out the line. Hyphenation slows down text layout and increases memory usage, so it should be used sparingly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHyphenationFactor:` (page 1503)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSLayoutManager.h`

init

Initializes the receiver, a newly created `NSLayoutManager` object.

```
- (id)init
```

Discussion
This method is the designated initializer for the `NSLayoutManager` class. Returns an initialized object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addLayoutManager:` (page 2837) (NSTextStorage)
- `addTextContainer:` (page 1449)

Declared In
`NSLayoutManager.h`

insertGlyph:atGlyphIndex:characterIndex:

Inserts a single glyph into the glyph stream at the given index and maps it to the character at the given character index.

```
- (void)insertGlyph:(NSGlyph)glyph atGlyphIndex:(NSUInteger)glyphIndex
    characterIndex:(NSUInteger)charIndex
```

Parameters

glyph
The glyph to insert.

glyphIndex
The index at which to insert the glyph.

charIndex
The index of the character to which the glyph is mapped.

Discussion
If the glyph is mapped to several characters, *charIndex* should indicate the first character it's mapped to.

This method is for use by the glyph-generation mechanism and doesn't perform any invalidation or generation of the glyphs or layout. This method should be invoked only during glyph generation and typesetting, in almost all cases only by the glyph generator or typesetter. For example, a custom glyph generator or typesetter might invoke it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `deleteGlyphsInRange:` (page 1457)
- `replaceGlyphAtIndex:withGlyph:` (page 1495)

Declared In
`NSLayoutManager.h`

insertGlyphs:length:forStartingGlyphAtIndex:characterIndex:

Inserts the given glyphs into the glyph cache at the given index and maps them to characters beginning at the given character index.

```
- (void)insertGlyphs:(const NSGlyph *)glyphs length:(NSUInteger)length
    forStartingGlyphAtIndex:(NSUInteger)glyphIndex
    characterIndex:(NSUInteger)charIndex
```

Parameters

glyphs
The glyphs to insert.

glyphIndex
The index in the glyph cache to begin inserting glyphs.

length
The number of glyphs to insert.

charIndex
Index of first character to be mapped.

Discussion
This method is part of the `NSGlyphStorage` protocol, for use by the glyph generator. It enables bulk insertion of glyphs into the glyph cache.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSLayoutManager.h`

insertTextContainer:atIndex:

Inserts the given text container into the series of text containers at the given index.

```
- (void)insertTextContainer:(NSTextContainer *)aTextContainer
    atIndex:(NSUInteger)index
```

Parameters

aTextContainer
The text container to insert.

index
The index in the series of text containers at which to insert *aTextContainer*.

Discussion
This method invalidates layout for all subsequent `NSTextContainer` objects, and invalidates glyph information as needed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addTextContainer:` (page 1449)
- `removeTextContainerAtIndex:` (page 1495)
- `textContainers` (page 1520)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSLayoutManager.h`

intAttribute:forGlyphAtIndex:

Returns the value of the attribute identified by the given attribute tag for the glyph at the given index.

```
- (NSInteger)intAttribute:(NSInteger)attributeTag
    forGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

attributeTag

The attribute whose value is returned.

glyphIndex

Index of the glyph whose attribute value is returned.

Return Value

The value of the attribute identified by *attributeTag* and *glyphIndex*.

Discussion

Subclasses that define their own custom attributes must override this method to access their own storage for the attribute values. Nonnegative tags are reserved by Apple; you can define your own attributes with negative tags and set values using `setIntAttribute:value:forGlyphAtIndex:` (page 1503).

If noncontiguous layout is not enabled, this method causes generation of all glyphs up to and including *glyphIndex*. This method is primarily for the use of the glyph generator and typesetter.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setIntAttribute:value:forGlyphAtIndex:` (page 1503)

Declared In

`NSLayoutManager.h`

invalidateDisplayForCharacterRange:

Invalidates display for the given character range.

```
- (void)invalidateDisplayForCharacterRange:(NSRange)charRange
```

Parameters

charRange

The character range for which display is invalidated.

Discussion

Parts of the range that are not laid out are remembered and redisplayed later when the layout is available. Does not actually cause layout.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSLayoutManager.h`

invalidateDisplayForGlyphRange:

Marks the glyphs in the given glyph range as needing display, as well as the appropriate regions of the `NSTextView` objects that display those glyphs (using the `NSView` method `setNeedsDisplayInRect:` (page 3225)).

```
- (void)invalidateDisplayForGlyphRange:(NSRange)glyphRange
```

Parameters

glyphRange
The range of glyphs to invalidate.

Discussion

You should rarely need to invoke this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSLayoutManager.h`

invalidateGlyphsForCharacterRange:changeInLength:actualCharacterRange:

Invalidates the cached glyphs for the characters in the given character range, adjusts the character indices of all the subsequent glyphs by the change in length, and invalidates the new character range.

```
- (void)invalidateGlyphsForCharacterRange:(NSRange)charRange
    changeInLength:(NSInteger)lengthChange
    actualCharacterRange:(NSRangePointer)actualCharRange
```

Parameters

charRange
The range of characters for which to invalidate glyphs.

lengthChange
The number of characters added or removed.

actualCharRange
If not `NULL`, on output, the actual range invalidated after any necessary expansion. This range can be larger than the range of characters given due to the effect of context on glyphs and layout.

Discussion

This method only invalidates glyph information and performs no glyph generation or layout. Because invalidating glyphs also invalidates layout, after invoking this method you should also invoke `invalidateLayoutForCharacterRange:actualCharacterRange:` (page 1483), passing *charRange* as the first argument.

This method is used by the layout mechanism and should be invoked only during typesetting, in almost all cases only by the typesetter. For example, a custom typesetter might invoke it.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSLayoutManager.h`

invalidateGlyphsOnLayoutInvalidationForGlyphRange:

Specifies explicitly when portions of the glyph stream depend on layout.

```
- (void)invalidateGlyphsOnLayoutInvalidationForGlyphRange:(NSRange)glyphRange
```

Parameters

glyphRange

The range of glyphs to invalidate.

Discussion

This method is for the use of the typesetter, to allow it to specify explicitly when portions of the glyph stream depend on layout, for example, because they have had hyphens inserted. Therefore, the glyphs are invalidated the next time their layout is invalidated, so that they will be regenerated before being laid out again.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSLayoutManager.h

invalidateLayoutForCharacterRange:actualCharacterRange:

Invalidates the layout information for the glyphs mapped to the given range of characters.

```
- (void)invalidateLayoutForCharacterRange:(NSRange)charRange
    actualCharacterRange:(NSRangePointer)actualCharRange
```

Parameters

charRange

The range of characters to invalidate.

actualCharRange

If not NULL, on output, the actual range invalidated after any necessary expansion.

Discussion

This method has the same effect as `invalidateLayoutForCharacterRange:isSoft:actualCharacterRange:` (page 1483) with *flag* set to NO.

This method only invalidates information; it performs no glyph generation or layout. You should rarely need to invoke this method.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- invalidateGlyphsForCharacterRange:changeInLength:actualCharacterRange:` (page 1482)

Declared In

NSLayoutManager.h

invalidateLayoutForCharacterRange:isSoft:actualCharacterRange:

Invalidates the layout information for the glyphs mapped to the given range of characters.

```
- (void)invalidateLayoutForCharacterRange:(NSRange)charRange isSoft:(BOOL)flag
    actualCharacterRange:(NSRangePointer)actualCharRange
```

Parameters

charRange

The character range for which glyphs are invalidated.

flag

If YES, invalidates internal caches in the layout manager; if NO, invalidates layout. See the discussion section.

actualCharRange

If not NULL, on output, the range of characters mapped to the glyphs whose layout information is invalidated. This range can be larger than the range of characters given due to the effect of context on glyphs and layout.

Discussion

This method only invalidates information; it performs no glyph generation or layout. You should rarely need to invoke this method.

For code that needs to work on both Mac OS X v10.5 and previous releases, the following procedures should be used. For Mac OS X v10.4 and before, invalidation should consist of

1. Calling this method with the *flag* set to YES, for the range that has actually become invalid.
2. Calling this method with the *flag* set to NO, for the range (if any) that follows that range, usually extending to the end of the text, that might need to be moved due to relayout of the invalidated range.

As of Mac OS X v10.5, the semantics of the *flag* parameter are slightly different. Soft layout holes are obsolete in Mac OS X v10.5 and later, so the flag is no longer necessary. If the method is called with *flag* set to NO, then it has the effect of invalidating layout. If it's called with the *flag* set to YES, then it does not actually invalidate layout; it invalidates a number of internal caches, but otherwise has no effect, and in general is unnecessary.

This method is superseded by `invalidateLayoutForCharacterRange:actualCharacterRange:` (page 1483) and will be deprecated in a future release.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- invalidateGlyphsForCharacterRange:changeInLength:actualCharacterRange:` (page 1482)

Declared In

NSLayoutManager.h

isValidGlyphIndex:

Indicates whether the specified index refers to a valid glyph, otherwise NO.

```
- (BOOL)isValidGlyphIndex:(NSUInteger)glyphIndex
```

Parameters

glyphIndex

The index of a glyph in the receiver.

Return Value

YES if the specified *glyphIndex* refers to a valid glyph, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSLayoutManager.h

layoutManagerOwnsFirstResponderInWindow:

Indicates whether the first responder in the given window is a text view associated with the receiver.

```
- (BOOL)layoutManagerOwnsFirstResponderInWindow:(NSWindow *)window
```

Parameters

window

The window whose first responder is tested.

Return Value

YES if the first responder in *window* is a text view associated with the receiver; otherwise, NO.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSLayoutManager.h

layoutOptions

Returns the layout manager's current layout options.

```
- (NSUInteger)layoutOptions
```

Return Value

A bit mask representing the current layout options as defined in Layout Options (page 3689) in *NSGlyphStorage Protocol Reference*.

Discussion

This method is part of the NSGlyphStorage protocol, for use by the glyph generator. It enables the glyph generator to ask which options the layout manager requests.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSLayoutManager.h

layoutRectForTextBlock:atIndex:effectiveRange:

Returns the layout rectangle within which the given text block containing the glyph at the given index is to be laid out.

```
- (NSRect)layoutRectForTextBlock:(NSTextBlock *)block atIndex:(NSUInteger)glyphIndex
    effectiveRange:(NSRangePointer)effectiveGlyphRange
```

Parameters

block

The text block whose layout rectangle is returned.

glyphIndex

Index of the glyph.

effectiveGlyphRange

If not `NULL`, on output, the range for all glyphs in the text block.

Return Value

The layout rectangle of the text block, or `NSZeroRect` if no rectangle has been set for the specified block since the last invalidation.

Discussion

This method causes glyph generation but not layout. Block layout rectangles and bounds rectangles are always in container coordinates.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- setLayoutRect:forTextBlock:glyphRange:` (page 1504)

Declared In

`NSLayoutManager.h`

layoutRectForTextBlock:glyphRange:

Returns the layout rectangle within which the given text block containing the given glyph range is to be laid out.

```
- (NSRect)layoutRectForTextBlock:(NSTextBlock *)block glyphRange:(NSRange)glyphRange
```

Return Value

The layout rectangle, or `NSZeroRect` if no rectangle has been set for the specified block since the last invalidation.

Discussion

This method causes glyph generation but not layout. Block layout rectangles and bounds rectangles are always in container coordinates.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- setLayoutRect:forTextBlock:glyphRange:` (page 1504)

Declared In
NSLayoutManager.h

lineFragmentRectForGlyphAtIndex:effectiveRange:

Returns the rectangle for the line fragment in which the given glyph is laid out and (optionally), by reference, the whole range of glyphs that are in that fragment.

```
- (NSRect)lineFragmentRectForGlyphAtIndex:(NSUInteger)glyphIndex
    effectiveRange:(NSRangePointer)effectiveGlyphRange
```

Parameters

glyphIndex
The glyph for which to return the line fragment rectangle.

effectiveGlyphRange
If not NULL, on output, the range for all glyphs in the line fragment.

Return Value
The line fragment in which the given glyph is laid out.

Discussion
This method causes glyph generation and layout for the line fragment containing the specified glyph, or if noncontiguous layout is not enabled, for all of the text up to and including that line fragment.

Line fragment rectangles are always in container coordinates.

Overriding this method is not recommended. If the the line fragment rectangle needs to be modified, that should be done at the typesetter level or by calling `setLineFragmentRect:forGlyphRange:usedRect:` (page 1505).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lineFragmentUsedRectForGlyphAtIndex:effectiveRange:` (page 1488)
- `setLineFragmentRect:forGlyphRange:usedRect:` (page 1505)

Related Sample Code
CircleView

Declared In
NSLayoutManager.h

lineFragmentRectForGlyphAtIndex:effectiveRange:withoutAdditionalLayout:

Returns the line fragment rectangle containing the glyph at the given glyph index.

```
- (NSRect)lineFragmentRectForGlyphAtIndex:(NSUInteger)glyphIndex
    effectiveRange:(NSRangePointer)effectiveGlyphRange
    withoutAdditionalLayout:(BOOL)flag
```

Parameters

glyphIndex

The glyph for which to return the line fragment rectangle.

effectiveGlyphRange

If not NULL, on output, the range for all glyphs in the line fragment.

flag

If YES, glyph generation and layout are not performed, so this option should not be used unless layout is known to be complete for the range in question, or unless noncontiguous layout is enabled; if NO, both are performed as needed.

Return Value

The line fragment in which the given glyph is laid out.

Discussion

This method is primarily for use from within NSTypesetter, after layout is complete for the range in question, but before the layout manager's call to NSTypesetter has returned. In that case glyph and layout holes have not yet been recalculated, so the layout manager does not yet know that layout is complete for that range, and this variant must be used.

Overriding this method is not recommended. If the the line fragment rectangle needs to be modified, that should be done at the typesetter level or by calling setLineFragmentRect:forGlyphRange:usedRect: (page 1505).

Availability

Available in Mac OS X v10.4 and later.

See Also

- setLineFragmentRect:forGlyphRange:usedRect: (page 1505)
- lineFragmentUsedRectForGlyphAtIndex:effectiveRange:withoutAdditionalLayout: (page 1489)

Declared In

NSLayoutManager.h

lineFragmentUsedRectForGlyphAtIndex:effectiveRange:

Returns the usage rectangle for the line fragment in which the given glyph is laid and (optionally) by reference the whole range of glyphs that are in that fragment.

```
- (NSRect)lineFragmentUsedRectForGlyphAtIndex:(NSUInteger)glyphIndex
    effectiveRange:(NSRangePointer)effectiveGlyphRange
```

Parameters

glyphIndex

The glyph for which to return the line fragment used rectangle.

effectiveGlyphRange

If not NULL, on output, the range for all glyphs in the line fragment.

Return Value

The used rectangle for the line fragment in which the given glyph is laid out.

Discussion
This method causes glyph generation and layout for the line fragment containing the specified glyph, or if noncontiguous layout is not enabled, up to and including that line fragment.

Line fragment used rectangles are always in container coordinates.

Overriding this method is not recommended. If the the line fragment used rectangle needs to be modified, that should be done at the typesetter level or by calling `setLineFragmentRect:forGlyphRange:usedRect:` (page 1505).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lineFragmentRectForGlyphAtIndex:effectiveRange:` (page 1487)
- `setLineFragmentRect:forGlyphRange:usedRect:` (page 1505)

Declared In
`NSLayoutManager.h`

lineFragmentUsedRectForGlyphAtIndex:effectiveRange:withoutAdditionalLayout:

Returns the usage rectangle for the line fragment in which the given glyph is laid and (optionally) by reference the whole range of glyphs that are in that fragment.

```
- (NSRect)lineFragmentUsedRectForGlyphAtIndex:(NSUInteger)glyphIndex
    effectiveRange:(NSRangePointer)effectiveGlyphRange
    withoutAdditionalLayout:(BOOL)flag
```

Parameters

glyphIndex
The glyph for which to return the line fragment used rectangle.

effectiveGlyphRange
If not `NULL`, on output, the range for all glyphs in the line fragment.

flag
If `YES`, glyph generation and layout are not performed, so this option should not be used unless layout is known to be complete for the range in question, or unless noncontiguous layout is enabled; if `NO`, both are performed as needed.

Return Value
The used rectangle for the line fragment in which the given glyph is laid out.

Discussion
This method causes glyph generation and layout for the line fragment containing the specified glyph, or if noncontiguous layout is not enabled, up to and including that line fragment.

Line fragment used rectangles are always in container coordinates.

Overriding this method is not recommended. If the the line fragment used rectangle needs to be modified, that should be done at the typesetter level or by calling `setLineFragmentRect:forGlyphRange:usedRect:` (page 1505).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setLineFragmentRect:forGlyphRange:usedRect:` (page 1505)
- `lineFragmentRectForGlyphAtIndex:effectiveRange:withoutAdditionalLayout:` (page 1487)

Declared In
`NSLayoutManager.h`

locationForGlyphAtIndex:

Returns the location for the given glyph within its line fragment.

```
- (NSPoint)locationForGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters
glyphIndex
The glyph whose location is returned.

Return Value
The location of the given glyph.

Discussion
If the given glyph does not have an explicit location set for it (for example, if it is part of (but not first in) a sequence of nominally spaced characters), the location is calculated by glyph advancements from the location of the most recent preceding glyph with a location set.

Glyph locations are relative to their line fragment rectangle's origin. The line fragment rectangle in turn is defined in the coordinate system of the text container where it resides.

This method causes glyph generation and layout for the line fragment containing the specified glyph, or if noncontiguous layout is not enabled, up to and including that line fragment.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lineFragmentRectForGlyphAtIndex:effectiveRange:` (page 1487)
- `lineFragmentUsedRectForGlyphAtIndex:effectiveRange:` (page 1488)

Related Sample Code
CircleView

Declared In
`NSLayoutManager.h`

notShownAttributeForGlyphAtIndex:

Indicates whether the glyph at the given index is one that isn't shown.

```
- (BOOL)notShownAttributeForGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters
glyphIndex
Index of the glyph.

Return Value
`YES` if the glyph at *`glyphIndex`* is not shown; otherwise `NO`.

Discussion
Some glyphs are not shown. For example, a tab, newline, or attachment glyph is not shown; it just affects the layout of following glyphs or locates the attachment graphic. Space characters, however, typically are shown as glyphs with a displacement, although they leave no visible marks.

This method causes glyph generation and layout for the line fragment containing the specified glyph, or if noncontiguous layout is not enabled, up to and including that line fragment.

Raises an `NSRangeException` if *`glyphIndex`* is out of bounds.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setNotShownAttribute:forGlyphAtIndex:` (page 1507)

Declared In
`NSLayoutManager.h`

numberOfGlyphs

Returns the number of glyphs in the receiver.

```
- (NSUInteger)numberOfGlyphs
```

Return Value
The number of glyphs.

Discussion
If noncontiguous layout is not enabled, this method forces generation of glyphs for all characters.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSLayoutManager.h`

rangeOfNominallySpacedGlyphsContainingIndex:

Returns the range for the glyphs around the given glyph that can be displayed using only their advancements from the font, without pairwise kerning or other adjustments to spacing.

```
- (NSRange)rangeOfNominallySpacedGlyphsContainingIndex:(NSUInteger)glyphIndex
```

Parameters
`glyphIndex`
Index of the glyph to test.

Return Value
The range of nominally spaced glyphs.

Discussion
The range returned begins with the first glyph, counting back from *glyphIndex*, that has a location set, and it continues up to, but does not include, the next glyph that has a location set.

Performs glyph generation and layout if needed.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSLayoutManager.h

rectArrayForCharacterRange:withinSelectedCharacterRange:inTextContainer:rectCount:

Returns an array of rectangles and, by reference, the number of such rectangles, that define the region in the given container enclosing the given character range.

```
- (NSRectArray)rectArrayForCharacterRange:(NSRange)charRange
    withinSelectedCharacterRange:(NSRange)selCharRange
    inTextContainer:(NSTextContainer *)container rectCount:(NSUInteger *)rectCount
```

Parameters

charRange
The character range for which to return rectangles.

selCharRange
Selected characters within *charRange*, which can affect the size of the rectangles; it must be equal to or contain *charRange*. If the caller is interested in this more from an enclosing point of view rather than a selection point of view, pass {NSNotFound, 0} as the selected range.

container
The text container in which the text is laid out.

rectCount
The number of rectangles returned.

Return Value
The array of rectangles enclosing the given range.

Discussion
These rectangles can be used to draw the text background or highlight for the given range of characters. If a selected range is given in *selCharRange*, the rectangles returned are correct for drawing the selection. Selection rectangles are generally more complicated than enclosing rectangles and supplying a selected range is the clue this method uses to determine whether to go to the trouble of doing this special work.

This method will do the minimum amount of work required to answer the question. The resulting array is owned by the layout manager and will be reused when this method, rectArrayForGlyphRange:withinSelectedGlyphRange:inTextContainer:rectCount: (page 1493), or boundingRectForGlyphRange:inTextContainer: (page 1452) is called. One of these methods may be called indirectly. If you aren't going to use the rectangles right away, you should copy them to another location. These rectangles are always in container coordinates.

The number of rectangles returned isn't necessarily the number of lines enclosing the specified range. Contiguous lines can share an enclosing rectangle, and lines broken into several fragments have a separate enclosing rectangle for each fragment.

These rectangles don't necessarily enclose glyphs that draw outside their line fragment rectangles; use `boundingRectForGlyphRange:inTextContainer:` (page 1452) to determine the area that contains all drawing performed for a range of glyphs.

Performs glyph generation and layout if needed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `glyphRangeForTextContainer:` (page 1477)
- `characterRangeForGlyphRange:actualGlyphRange:` (page 1455)
- `drawsOutsideLineFragmentForGlyphAtIndex:` (page 1459)

Declared In
`NSLayoutManager.h`

rectArrayForGlyphRange:withinSelectedGlyphRange:inTextContainer:rectCount:

Returns an array of rectangles and, by reference, the number of such rectangles, that define the region in the given container enclosing the given glyph range.

```
- (NSRectArray)rectArrayForGlyphRange:(NSRange)glyphRange
    withinSelectedGlyphRange:(NSRange)selGlyphRange inTextContainer:(NSTextContainer
     *)container rectCount:(NSUInteger *)rectCount
```

Parameters

glyphRange
The glyph range for which to return rectangles.

selGlyphRange
Selected glyphs within *glyphRange*, which can affect the size of the rectangles; it must be equal to or contain *glyphRange*. If the caller is interested in this more from an enclosing point of view rather than a selection point of view, pass `{NSNotFound, 0}` as the selected range.

container
The text container in which the text is laid out.

rectCount
The number of rectangles returned.

Return Value
The array of rectangles enclosing the given range.

Discussion
These rectangles can be used to draw the text background or highlight for the given range of characters. If a selected range is given in *selGlyphRange*, the rectangles returned are correct for drawing the selection. Selection rectangles are generally more complicated than enclosing rectangles and supplying a selected range is the clue this method uses to determine whether to go to the trouble of doing this special work.

The number of rectangles returned isn't necessarily the number of lines enclosing the specified range. Contiguous lines can share an enclosing rectangle, and lines broken into several fragments have a separate enclosing rectangle for each fragment.

This method will do the minimum amount of work required to answer the question. The resulting array is owned by the layout manager and will be reused when this method, `rectArrayForCharacterRange:withinSelectedCharacterRange:inTextContainer:rectCount:` (page 1492), or `boundingRectForGlyphRange:inTextContainer:` (page 1452) is called. One of these methods may be called indirectly. If you aren't going to use the rectangles right away, you should copy them to another location. These rectangles are always in container coordinates.

The purpose of this method is to calculate line rectangles for drawing the text background and highlighting. These rectangles don't necessarily enclose glyphs that draw outside their line fragment rectangles; use `boundingRectForGlyphRange:inTextContainer:` (page 1452) to determine the area that contains all drawing performed for a range of glyphs.

Performs glyph generation and layout if needed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `glyphRangeForTextContainer:` (page 1477)
- `glyphRangeForCharacterRange:actualCharacterRange:` (page 1476)
- `drawsOutsideLineFragmentForGlyphAtIndex:` (page 1459)

Declared In
`NSLayoutManager.h`

removeTemporaryAttribute:forCharacterRange:

Removes a temporary attribute from the list of attributes for the specified character range.

```
- (void)removeTemporaryAttribute:(NSString *)attrName
    forCharacterRange:(NSRange)charRange
```

Parameters

attrName
The name of a temporary attribute.

charRange
The range of characters from which to remove the specified temporary attribute.

Discussion
Temporary attributes are used only for onscreen drawing and are not persistent in any way. `NSTextView` uses them to color misspelled words when continuous spell checking is enabled. Currently the only temporary attributes recognized are those that do not affect layout (colors, underlines, and so on).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTemporaryAttributes:forCharacterRange:` (page 1508)
- `addTemporaryAttributes:forCharacterRange:` (page 1449)
- `temporaryAttributesAtCharacterIndex:effectiveRange:` (page 1517)

Related Sample Code
LayoutManagerDemo

Declared In
NSLayoutManager.h

removeTextContainerAtIndex:

Removes the text container at the given index and invalidates the layout as needed.

```
- (void)removeTextContainerAtIndex:(NSUInteger)index
```

Parameters

index
The index of the text container to remove.

Discussion
This method invalidates glyph information as needed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- addTextContainer: (page 1449)
- insertTextContainer:atIndex: (page 1480)
- textContainers (page 1520)
- invalidateGlyphsForCharacterRange:changeInLength:actualCharacterRange: (page 1482)
- invalidateLayoutForCharacterRange:isSoft:actualCharacterRange: (page 1483)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSLayoutManager.h

replaceGlyphAtIndex:withGlyph:

Replaces the glyph at the given index with a new glyph.

```
- (void)replaceGlyphAtIndex:(NSUInteger)glyphIndex withGlyph:(NSGlyph)newGlyph
```

Parameters

glyphIndex
Index of the glyph to replace.

newGlyph
The new glyph.

Discussion
Doesn't alter the glyph-to-character mapping or invalidate layout information. The character index of the glyph is assumed to remain the same (although it can, of course, be set explicitly if needed).

This method is for use by the glyph-generation mechanism and doesn't perform any invalidation or generation of the glyphs or layout. This method should be invoked only during glyph generation and typesetting, in almost all cases only by the glyph generator or typesetter. For example, a custom glyph generator or typesetter might invoke it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setCharacterIndex:forGlyphAtIndex:` (page 1500)
- `invalidateGlyphsForCharacterRange:changeInLength:actualCharacterRange:` (page 1482)
- `invalidateLayoutForCharacterRange:isSoft:actualCharacterRange:` (page 1483)

Declared In
`NSLayoutManager.h`

replaceTextStorage:

Replaces the `NSTextStorage` object for the group of text-system objects containing the receiver with the given text storage object.

```
- (void)replaceTextStorage:(NSTextStorage *)newTextStorage
```

Parameters

newTextStorage
The text storage object to set.

Discussion
All `NSLayoutManager` objects sharing the original `NSTextStorage` object then share the new one. This method makes all the adjustments necessary to keep these relationships intact, unlike `setTextStorage:` (page 1509).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
TextLayoutDemo

Declared In
`NSLayoutManager.h`

rulerAccessoryViewForTextView:paragraphStyle:ruler:enabled:

Returns the the accessory view that the text system uses for its ruler.

```
- (NSView *)rulerAccessoryViewForTextView:(NSTextView *)view
    paragraphStyle:(NSParagraphStyle *)style ruler:(NSRulerView *)ruler
    enabled:(BOOL)isEnabled
```

Parameters

view
The text view using the layout manager.

style
Sets the state of the controls in the accessory view; must not be `nil`.

ruler
The ruler view whose accessory view is returned.

isEnabled
If YES, the accessory view is enabled and accepts mouse and keyboard events; if NO it's disabled.

Return Value
The accessory view containing tab wells, text alignment buttons, and so on.

Discussion
If you have turned off automatic ruler updating through the use of setUsesRuler: (page 2948) so that you can do more complex things, but you still want to display the appropriate accessory view, you can use this method.

This method is invoked automatically by the NSTextView object using the layout manager. You should rarely need to invoke it, but you can override it to customize ruler support. If you do use this method directly, note that it neither installs the ruler accessory view nor sets the markers for the NSRulerView object. You must install the accessory view into the ruler using the NSRulerView method setAccessoryView: (page 2258). To set the markers, use rulerMarkersForTextView:paragraphStyle:ruler: (page 1497) to get the markers needed, and then send setMarkers: (page 2259) to the ruler.

Availability
Available in Mac OS X v10.0 and later.

See Also
- horizontalRulerView (page 2350) (NSScrollView)

Declared In
NSLayoutManager.h

rulerMarkersForTextView:paragraphStyle:ruler:

Returns an array of text ruler objects for the current selection.

```
- (NSArray *)rulerMarkersForTextView:(NSTextView *)view
    paragraphStyle:(NSParagraphStyle *)style ruler:(NSRulerView *)ruler
```

Parameters

view
The text view using the layout manager.

style
Sets the state of the controls in the accessory view; must not be nil.

ruler
The ruler view whose ruler markers are returned.

Return Value
An array of NSRulerMarker objects representing such things as left and right margins, first-line indent, and tab stops.

Discussion
If you have turned off automatic ruler updating through the use of setUsesRuler: (page 2948) so that you can do more complex things, but you still want to display the appropriate accessory view, you can use this method.

This method is invoked automatically by the NSTextView object using the layout manager. You should rarely need to invoke it, but you can override it to add new kinds of markers or otherwise customize ruler support.

You can set the returned ruler markers with the `NSRulerView` method `setMarkers:` (page 2259).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- rulerAccessoryViewForTextView:paragraphStyle:ruler:enabled:` (page 1496)

Declared In
`NSLayoutManager.h`

setAllowsNonContiguousLayout:

Enables or disables noncontiguous layout.

```
- (void)setAllowsNonContiguousLayout:(BOOL)flag
```

Parameters

flag

If `YES`, noncontiguous layout is enabled; if `NO`, noncontiguous layout is disabled.

Discussion
Passing `YES` in *flag* allows but does not require the layout manager to use noncontiguous layout, and the layout manager may in fact not do so, depending on its configuration.

For more information about noncontiguous layout, see "Noncontiguous Layout" (page 1438).

Availability
Available in Mac OS X v10.5 and later.

See Also
`- allowsNonContiguousLayout` (page 1450)
`- hasNonContiguousLayout` (page 1477)

Declared In
`NSLayoutManager.h`

setAttachmentSize:forGlyphRange:

Sets the size at which the given glyph (assumed to be an attachment) is asked to draw in the given glyph range.

```
- (void)setAttachmentSize:(NSSize)attachmentSize forGlyphRange:(NSRange)glyphRange
```

Parameters

attachmentSize

The glyph size to set.

glyphRange

The attachment glyph's position in the glyph stream.

Discussion
For a glyph corresponding to an attachment, this method should be called to set the size for the attachment cell to occupy. The glyph's value should be `NSControlGlyph`.

This method is used by the layout mechanism and should be invoked only during typesetting, in almost all cases only by the typesetter. For example, a custom typesetter might invoke it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `attachmentSizeForGlyphAtIndex:` (page 1450)
- `setDefaultAttachmentScaling:` (page 1500)

Declared In
`NSLayoutManager.h`

setBackgroundLayoutEnabled:

Specifies whether the receiver generates glyphs and lays them out when the application's run loop is idle.

```
- (void)setBackgroundLayoutEnabled:(BOOL)flag
```

Parameters

flag
If `YES`, background layout is enabled; if `NO`, the receiver performs glyph generation and layout only when necessary.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `backgroundLayoutEnabled` (page 1451)

Declared In
`NSLayoutManager.h`

setBoundsRect:forTextBlock:glyphRange:

Sets the bounding rectangle enclosing a given text block containing the given glyph range.

```
- (void)setBoundsRect:(NSRect)rect forTextBlock:(NSTextBlock *)block
    glyphRange:(NSRange)glyphRange
```

Parameters

rect
The bounding rectangle to set.

block
The text block whose bounding rectangle is set.

glyphRange
The range of glyphs in the text block.

Discussion
This method causes glyph generation but not layout. Block layout rectangles and bounds rectangles are always in container coordinates.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `boundingRectForGlyphRange:inTextContainer:` (page 1452)
- `boundsRectForTextBlock:atIndex:effectiveRange:` (page 1452)
- `boundsRectForTextBlock:glyphRange:` (page 1453)

Declared In
`NSLayoutManager.h`

setCharacterIndex:forGlyphAtIndex:

Sets the index of the character corresponding to the glyph at the given glyph index.

```
- (void)setCharacterIndex:(NSUInteger)charIndex
    forGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

charIndex
The index to set.

glyphIndex
The glyph corresponding to the character whose index is set. The glyph must already be present.

Discussion
This method is for use by the glyph-generation mechanism and doesn't perform any invalidation or generation of the glyphs or layout. This method should be invoked only during glyph generation and typesetting, in almost all cases only by the glyph generator or typesetter. For example, a custom glyph generator or typesetter might invoke it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `characterIndexForGlyphAtIndex:` (page 1453)
- `characterRangeForGlyphRange:actualGlyphRange:` (page 1455)
- `glyphRangeForCharacterRange:actualCharacterRange:` (page 1476)

Declared In
`NSLayoutManager.h`

setDefaultAttachmentScaling:

Sets the default scaling behavior to the given scaling if an attachment image is too large to fit in a text container.

```
- (void)setDefaultAttachmentScaling:(NSImageScaling)scaling
```

Parameters

scaling
The scaling behavior to set. See `NSImageScaling` (page 617) for possible values. The default is `NSScaleNone` (page 622), meaning that images clip rather than scaling.

Discussion
Attachment cells control their own size and drawing, so this setting is only advisory to them, but Application Kit–supplied attachment cells respect it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `defaultAttachmentScaling` (page 1456)

Declared In
`NSLayoutManager.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSLayoutManagerDelegate >)anObject
```

Parameters

anObject
The delegate for the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `delegate` (page 1457)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSLayoutManager.h`

setDrawsOutsideLineFragment:forGlyphAtIndex:

Specifies whether the given glyph exceeds the bounds of the line fragment where it's laid out.

```
- (void)setDrawsOutsideLineFragment:(BOOL)flag forGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

flag
If `YES`, sets the given glyph to draw outside its line fragment; if `NO`, the glyph does not draw outside.

glyphIndex
Index of the glyph to set.

Discussion
This can happen when text is set at a fixed line height. For example, if the user specifies a fixed line height of 12 points and sets the font size to 24 points, the glyphs will exceed their layout rectangles. This information is important for determining whether additional lines need to be redrawn as a result of changes to any given line fragment.

This method is used by the layout mechanism and should be invoked only during typesetting, in almost all cases only by the typesetter. For example, a custom typesetter might invoke it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawsOutsideLineFragmentForGlyphAtIndex:` (page 1459)

Declared In
`NSLayoutManager.h`

setExtraLineFragmentRect:usedRect:textContainer:

Sets the bounds and container for the extra line fragment.

```
- (void)setExtraLineFragmentRect:(NSRect)aRect usedRect:(NSRect)usedRect
    textContainer:(NSTextContainer *)aTextContainer
```

Parameters

aRect
The rectangle to set.

usedRect
Indicates where the insertion point is drawn.

aTextContainer
The text container where the rectangle is to be laid out.

Discussion
The extra line fragment is used when the text backing ends with a hard line break or when the text backing is totally empty, to define the extra line which needs to be displayed at the end of the text. If the text backing is not empty and does not end with a hard line break, this should be set to `NSZeroRect` and `nil`.

Line fragment rectangles and line fragment used rectangles are always in container coordinates.

This method is used by the layout mechanism and should be invoked only during typesetting, in almost all cases only by the typesetter. For example, a custom typesetter might invoke it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `extraLineFragmentRect` (page 1464)
- `extraLineFragmentUsedRect` (page 1465)
- `textContainers` (page 1520)

Declared In
`NSLayoutManager.h`

setGlyphGenerator:

Sets the glyph generator used by this layout manager.

```
- (void)setGlyphGenerator:(NSGlyphGenerator *)glyphGenerator
```

Parameters

glyphGenerator

The new glyph generator to set.

Discussion

Setting the glyph generator invalidates all glyphs and layout in the layout manager.

Availability

Available in Mac OS X v10.4 and later.

See Also

- glyphGenerator (page 1472)

Declared In

NSLayoutManager.h

setHyphenationFactor:

Sets the threshold controlling when hyphenation is done.

```
- (void)setHyphenationFactor:(float)factor
```

Parameters

factor

The hyphenation factor, ranging from 0.0 to 1.0. By default, the value is 0.0, meaning hyphenation is off. A *factor* of 1.0 causes hyphenation to be attempted always.

Discussion

Whenever (width of the real contents of the line) / (the line fragment width) is below *factor*, hyphenation is attempted when laying out the line. Hyphenation slows down text layout and increases memory usage, so it should be used sparingly.

May be overridden on a per-paragraph basis by the NSParagraphStyle method hyphenationFactor (page 1873).

Availability

Available in Mac OS X v10.0 and later.

See Also

- hyphenationFactor (page 1478)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

NSLayoutManager.h

setIntAttribute:value:forGlyphAtIndex:

Sets a custom attribute value for a given glyph.

```
- (void)setIntAttribute:(NSInteger)attributeTag value:(NSInteger)val
    forGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

attributeTag

The custom attribute.

val

The new attribute value.

glyphIndex

Index of the glyph whose attribute is set.

Discussion

Custom attributes are glyph attributes such as `NSGlyphInscription` or attributes defined by subclasses. Nonnegative tags are reserved by Apple; you can define your own attributes with negative tags and set values using this method.

This method is part of the `NSGlyphStorage` protocol, for use by the glyph generator to set attributes. It is not usually necessary for anyone but the glyph generator (and perhaps the typesetter) to call it. It is provided as a public method so subclasses can extend it to accept other glyph attributes. To add new glyph attributes to the text system you must do two things. First, you need to arrange for the glyph generator or typesetter to generate and interpret it. Second, you need to subclass `NSLayoutManager` to provide someplace to store the new attribute, overriding this method and `intAttribute:forGlyphAtIndex:` (page 1481) to recognize the new attribute tags and respond to them, while passing any other attributes to the superclass implementation. The `NSLayoutManager` implementation understands the glyph attributes which it is prepared to store, as enumerated in "Glyph Attributes" (page 1525).

Availability

Available in Mac OS X v10.0 and later.

See Also

`- intAttribute:forGlyphAtIndex:` (page 1481)

Declared In

`NSLayoutManager.h`

setLayoutRect:forTextBlock:glyphRange:

Sets the layout rectangle enclosing the given text block containing the given glyph range.

```
- (void)setLayoutRect:(NSRect)rect forTextBlock:(NSTextBlock *)block
    glyphRange:(NSRange)glyphRange
```

Parameters

rect

The layout rectangle to set.

block

The text block whose layout rectangle is set.

glyphRange

The range of glyphs in the text block.

Discussion

This method causes glyph generation but not layout. Block layout rectangles and bounds rectangles are always in container coordinates.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `layoutRectForTextBlock:atIndex:effectiveRange:` (page 1486)
- `layoutRectForTextBlock:glyphRange:` (page 1486)

Declared In
`NSLayoutManager.h`

setLineFragmentRect:forGlyphRange:usedRect:

Associates the given line fragment bounds with the given range of glyphs.

```
- (void)setLineFragmentRect:(NSRect)fragmentRect forGlyphRange:(NSRange)glyphRange
    usedRect:(NSRect)usedRect
```

Parameters

fragmentRect
The rectangle of the line fragment.

glyphRange
The range of glyphs to be associated with *fragmentRect*.

usedRect
The portion of *fragmentRect* that actually contains glyphs or other marks that are drawn (including the text container's line fragment padding. Must be equal to or contained within *fragmentRect*.

Discussion
The typesetter must specify the text container first with `setTextContainer:forGlyphRange:` (page 1509), and it sets the exact positions of the glyphs afterwards with `setLocation:forStartOfGlyphRange:` (page 1506).

In the course of layout, all glyphs should end up being included in a range passed to this method, but only glyphs that start a new line fragment should be at the start of such ranges.

Line fragment rectangles and line fragment used rectangles are always in container coordinates.

This method is used by the layout mechanism and should be invoked only during typesetting, in almost all cases only by the typesetter. For example, a custom typesetter might invoke it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lineFragmentRectForGlyphAtIndex:effectiveRange:withoutAdditionalLayout:` (page 1487)
- `lineFragmentRectForGlyphAtIndex:effectiveRange:` (page 1487)
- `lineFragmentUsedRectForGlyphAtIndex:effectiveRange:withoutAdditionalLayout:` (page 1489)
- `lineFragmentUsedRectForGlyphAtIndex:effectiveRange:` (page 1488)

Declared In
`NSLayoutManager.h`

setLocation:forStartOfGlyphRange:

Sets the location for the first glyph of the given range.

```
- (void)setLocation:(NSPoint)aPoint forStartOfGlyphRange:(NSRange)glyphRange
```

Parameters

aPoint

The location to which the first glyph is set, relative to the origin of the glyph's line fragment origin.

glyphRange

The glyphs whose location is set.

Discussion

Setting the location for a glyph range implies that its first glyph is not nominally spaced with respect to the previous glyph. In the course of layout, all glyphs should end up being included in a range passed to this method, but only glyphs that start a new nominal range should be at the start of such ranges. The first glyph in a line fragment should always start a new nominal range. Glyph locations are given relative to their line fragment rectangle's origin.

Before setting the location for a glyph range, you must specify the text container with `setTextContainer:forGlyphRange:` (page 1509) and the line fragment rectangle with `setLineFragmentRect:forGlyphRange:usedRect:` (page 1505).

This method is used by the layout mechanism and should be invoked only during typesetting, in almost all cases only by the typesetter. For example, a custom typesetter might invoke it.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- rangeOfNominallySpacedGlyphsContainingIndex:` (page 1491)

Declared In

`NSLayoutManager.h`

setLocations:startingGlyphIndexes:count:forGlyphRange:

Sets locations for many glyph ranges at once.

```
- (void)setLocations:(NSPointArray)locations startingGlyphIndexes:(NSUInteger
    *)glyphIndexes count:(NSUInteger)count forGlyphRange:(NSRange)glyphRange
```

Parameters

locations

The locations to which the first glyph in each range is set, relative to the origin of the glyph's line fragment origin.

glyphIndexes

Indexes in *glyphRange* of the glyphs whose locations are set.

count

The number of glyphs whose locations are set.

glyphRange

The entire glyph range containing all the glyphs whose locations are set.

Discussion
This method enables the typesetter to set locations for glyph ranges in bulk. All of the specified glyph indexes should lie within the specified glyph range. The first of them should be equal to `glyphRange.location`, and the remainder should increase monotonically. Each location is set as the location for the range beginning at the corresponding glyph index, and continuing until the subsequent glyph index, or until the end of the glyph range for the last location. Thus this method is equivalent to calling `setLocation:forStartOfGlyphRange:` (page 1506) for a set of ranges covering all of the glyphs in *glyphRange*.

This method is used by the layout mechanism and should be invoked only during typesetting, in almost all cases only by the typesetter. For example, a custom typesetter might invoke it.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSLayoutManager.h`

setNotShownAttribute:forGlyphAtIndex:

Sets the glyph at the given index to be one that isn't shown.

```
- (void)setNotShownAttribute:(BOOL)flag forGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

flag
If `YES`, the glyph is not shown; if `NO`, it is shown.

glyphIndex
Index of the glyph whose attribute is set.

Discussion
The typesetter decides which glyphs are not shown and sets this attribute in the layout manager to ensure that those glyphs are not displayed. For example, a tab or newline character doesn't leave any marks; it just indicates where following glyphs are laid out.

Raises an `NSRangeException` if *glyphIndex* is out of bounds.

This method is used by the layout mechanism and should be invoked only during typesetting, in almost all cases only by the typesetter. For example, a custom typesetter might invoke it.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- notShownAttributeForGlyphAtIndex:` (page 1490)

Declared In
`NSLayoutManager.h`

setShowsControlCharacters:

Specifies whether to substitute visible glyphs for control characters in layout.

```
- (void)setShowsControlCharacters:(BOOL)flag
```

Parameters

flag

If YES, the receiver substitutes visible glyphs for control characters if the font and script support it; if NO, it doesn't. The default is NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setShowsInvisibleCharacters: (page 1508)
- showsControlCharacters (page 1513)

Declared In
NSLayoutManager.h

setShowsInvisibleCharacters:

Specifies whether to substitute visible glyphs for whitespace and other typically invisible characters in layout.

```
- (void)setShowsInvisibleCharacters:(BOOL)flag
```

Parameters

flag

If YES, the receiver substitutes visible glyphs for invisible characters if the font and script support it; if NO, it doesn't. The default is NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setShowsControlCharacters: (page 1507)
- showsInvisibleCharacters (page 1513)

Declared In
NSLayoutManager.h

setTemporaryAttributes:forCharacterRange:

Sets one or more temporary attributes for the specified character range.

```
- (void)setTemporaryAttributes:(NSDictionary *)attrs
    forCharacterRange:(NSRange)charRange
```

Parameters

attrs

Attributes dictionary containing the temporary attributes to set.

charRange

The range of characters to which the specified attributes apply.

Discussion
Temporary attributes are used only for onscreen drawing and are not persistent in any way. `NSTextView` uses them to color misspelled words when continuous spell checking is enabled. Currently the only temporary attributes recognized are those that do not affect layout (colors, underlines, and so on).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addTemporaryAttributes:forCharacterRange:` (page 1449)
- `removeTemporaryAttribute:forCharacterRange:` (page 1494)
- `temporaryAttributesAtCharacterIndex:effectiveRange:` (page 1517)

Declared In
`NSLayoutManager.h`

setTextContainer:forGlyphRange:

Sets text container where the glyphs in the given range are laid out.

```
- (void)setTextContainer:(NSTextContainer *)aTextContainer
    forGlyphRange:(NSRange)glyphRange
```

Parameters

aTextContainer
The text container to set.

glyphRange
The range of glyphs to lay out.

Discussion
The layout within the container is specified with the `setLineFragmentRect:forGlyphRange:usedRect:` (page 1505) and `setLocation:forStartOfGlyphRange:` (page 1506) methods.

This method is used by the layout mechanism and should be invoked only during typesetting, in almost all cases only by the typesetter. For example, a custom typesetter might invoke it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textContainerForGlyphAtIndex:effectiveRange:` (page 1519)

Declared In
`NSLayoutManager.h`

setTextStorage:

Sets the receiver's `NSTextStorage` object.

```
- (void)setTextStorage:(NSTextStorage *)textStorage
```

Parameters

textStorage

The text storage object to set.

Discussion

This method is invoked automatically when you add an `NSLayoutManager` to an `NSTextStorage` object; you should never need to invoke it directly, but you might want to override it. If you want to replace the `NSTextStorage` object for an established group of text-system objects containing the receiver, use `replaceTextStorage:` (page 1496).

Availability

Available in Mac OS X v10.0 and later.

See Also

`- addLayoutManager:` (page 2837) (`NSTextStorage`)

Declared In

`NSLayoutManager.h`

setTypesetter:

Sets the current typesetter.

```
- (void)setTypesetter:(NSTypesetter *)typesetter
```

Parameters

typesetter

The typesetter for the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- typesetter` (page 1522)

Declared In

`NSLayoutManager.h`

setTypesetterBehavior:

Sets the default typesetter behavior.

```
- (void)setTypesetterBehavior:(NSTypesetterBehavior)theBehavior
```

Parameters

theBehavior

An `NSTypesetterBehavior` (page 1527) constant that specifies the behavior for the receiver.

Discussion

The typesetter behavior affects glyph spacing and line height.

If the application was linked on a system prior to Mac OS X v10.2, `NSLayoutManager` uses `NSTypesetterOriginalBehavior` by default.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `typesetterBehavior` (page 1522)

Declared In
`NSLayoutManager.h`

setUsesFontLeading:

Specifies whether or not the receiver uses the leading provided in the font.

```
- (void)setUsesFontLeading:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver uses the font's leading; if `NO`, it does not.

Discussion
By default, a layout manager uses leading as specified by the font. However, this is not appropriate for most user-interface text, for which a fixed leading is usually specified by user-interface layout guidelines. This method enables the use of the font's leading to be turned off.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `usesFontLeading` (page 1524)
- `setLineSpacing:` (page 1722) (`NSMutableParagraphStyle`)

Declared In
`NSLayoutManager.h`

setUsesScreenFonts:

Controls using screen fonts to calculate layout and display text.

```
- (void)setUsesScreenFonts:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver uses screen fonts; if `NO`, it doesn't.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `usesScreenFonts` (page 1525)
- `substituteFontForFont:` (page 1514)

Related Sample Code
CircleView

TextLayoutDemo

Declared In
NSLayoutManager.h

showAttachmentCell:inRect:characterIndex:

Draws an attachment cell.

```
- (void)showAttachmentCell:(NSCell *)cell inRect:(NSRect)rect
    characterIndex:(NSUInteger)attachmentIndex
```

Parameters

cell
The attachment cell to draw.

rect
The rectangle within which to draw *cell*.

attachmentIndex
The location of the attachment cell.

Discussion
The *attachmentIndex* parameter is provided for cells that alter their appearance based on their location.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSLayoutManager.h

showPackedGlyphs:length:glyphRange:atPoint:font:color:printingAdjustment:

Draws a range of glyphs.

```
- (void)showPackedGlyphs:(char *)glyphs length:(NSUInteger)glyphLen
    glyphRange:(NSRange)glyphRange atPoint:(NSPoint)point font:(NSFont *)font
    color:(NSColor *)color printingAdjustment:(NSSize)printingAdjustment
```

Parameters

glyphs
The glyphs to draw; may contain embedded NULL bytes.

glyphLen
The number of bytes pointed at by *glyphs*; this is twice the number of glyphs contained.

glyphRange
The range of glyphs to draw.

point
The point at which to draw the glyphs.

font
The font of the glyphs to draw.

color
Color of the glyphs to draw.

printingAdjustment
NSZeroSize when drawing to the screen, but when printing may contain values by which the nominal spacing between the characters should be adjusted.

Discussion
The *glyphRange*, *point*, *font*, and *color* parameters are passed in merely for information purposes. They are already set in the graphics state. If for any reason you modify the set color or font, you must restore it before returning from this method.

You should never call this method, but you might override it.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSLayoutManager.h

showsControlCharacters

Indicates whether the receiver substitutes visible glyphs for control characters.

```
- (BOOL)showsControlCharacters
```

Return Value
YES if the receiver substitutes visible glyphs for control characters if the font and script support it; NO if it doesn't.

Availability
Available in Mac OS X v10.0 and later.

See Also
- showsInvisibleCharacters (page 1513)
- setShowsControlCharacters: (page 1507)

Declared In
NSLayoutManager.h

showsInvisibleCharacters

Indicates whether the receiver substitutes visible glyphs for whitespace and other typically invisible characters in layout.

```
- (BOOL)showsInvisibleCharacters
```

Return Value
YES if the receiver substitutes visible glyphs for invisible characters if the font and script support it; otherwise NO. The default is NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- showsControlCharacters (page 1513)

- setShowsInvisibleCharacters: (page 1508)

Declared In
NSLayoutManager.h

strikethroughGlyphRange:strikethroughType:lineFragmentRect: lineFragmentGlyphRange:containerOrigin:

Calculates and draws strikethrough for the glyphs in the given range.

```
- (void)strikethroughGlyphRange:(NSRange)glyphRange
    strikethroughType:(NSInteger)strikethroughVal lineFragmentRect:(NSRect)lineRect
     lineFragmentGlyphRange:(NSRange)lineGlyphRange
    containerOrigin:(NSPoint)containerOrigin
```

Parameters

glyphRange
The range of glyphs for which to draw a strikethrough. The range must belong to a single line fragment rectangle (as returned by lineFragmentRectForGlyphAtIndex:effectiveRange: (page 1487)).

strikethroughVal
The style of underlining to draw. This value is a mask derived from the value for NSUnderlineStyleAttributeName (page 272)—for example, (NSUnderlinePatternDash | NSUnderlineStyleThick | NSUnderlineByWordMask). Subclasses can define custom underlining styles.

lineRect
The line fragment rectangle containing the glyphs to draw strikethrough for.

lineGlyphRange
The range of all glyphs within *lineRect*.

containerOrigin
The origin of the line fragment rectangle's NSTextContainer in its NSTextView.

Discussion
This method determines which glyphs actually need to have a strikethrough drawn based on *strikethroughVal*. After determining which glyphs to draw strikethrough on, this method invokes drawStrikethroughForGlyphRange:strikethroughType:baselineOffset: lineFragmentRect:lineFragmentGlyphRange:containerOrigin: (page 1460) for each contiguous range of glyphs that requires it.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSLayoutManager.h

substituteFontForFont:

Returns a screen font suitable for use in place of the given font, if one is available.

```
- (NSFont *)substituteFontForFont:(NSFont *)originalFont
```

Parameters

originalFont

The font to replace.

Return Value

A screen font suitable for use in place of *originalFont*, or simply *originalFont* if a screen font can't be used or isn't available.

Discussion

A screen font can be substituted if the receiver is set to use screen fonts and if no `NSTextView` associated with the receiver is scaled or rotated.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- usesScreenFonts` (page 1525)

Declared In

`NSLayoutManager.h`

temporaryAttribute:atCharacterIndex:effectiveRange:

Returns the value for the temporary attribute with a given name of the character at a given index, and by reference the range over which the attribute applies.

```
- (id)temporaryAttribute:(NSString *)attrName atCharacterIndex:(NSUInteger)location
    effectiveRange:(NSRangePointer)range
```

Parameters

attrName

The name of a temporary attribute.

location

The index for which to return attributes. This value must not exceed the bounds of the receiver.

range

If non-`NULL`:

- If the named attribute exists at *location*, on output, contains the range over which the named attribute's value applies.
- If the named attribute does not exist at *location*, on output, contains the range over which the attribute does not exist.

The range isn't necessarily the maximum range covered by *attrName*, and its extent is implementation-dependent. If you need the maximum range, use `temporaryAttribute:atCharacterIndex:longestEffectiveRange:inRange:` (page 1516). If you don't need this value, pass `NULL`.

Return Value

The value for the temporary attribute named *attrName* of the character at index *location*, or `nil` if there is no such attribute.

Availability

Available in Mac OS X v10.5 and later.

See Also
- `temporaryAttributesAtCharacterIndex:effectiveRange:` (page 1517)
- `temporaryAttribute:atCharacterIndex:longestEffectiveRange:inRange:` (page 1516)

Declared In
`NSLayoutManager.h`

temporaryAttribute:atCharacterIndex:longestEffectiveRange:inRange:

Returns the value for the temporary attribute with a given name of the character at a given index, and by reference the maximum range over which the attribute applies.

```
- (id)temporaryAttribute:(NSString *)attrName atCharacterIndex:(NSUInteger)location
    longestEffectiveRange:(NSRangePointer)range inRange:(NSRange)rangeLimit
```

Parameters

attrName
The name of a temporary attribute.

location
The index for which to return attributes. This value must not exceed the bounds of the receiver.

range
If non-`NULL`:

- If the named attribute exists at *location*, on output, contains the maximum range over which the named attribute's value applies, clipped to *rangeLimit*.
- If the named attribute does not exist at *location*, on output, contains the maximum range over which the attribute does not exist.

If you don't need this value, pass `NULL`.

rangeLimit
The range over which to search for continuous presence of *attrName*. This value must not exceed the bounds of the receiver.

Return Value
The value for the attribute named *attrName* of the character at *location*, or `nil` if there is no such attribute.

Discussion
If you don't need the longest effective range, it's far more efficient to use the `temporaryAttribute:atCharacterIndex:effectiveRange:` (page 1515) method to retrieve the attribute value.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `temporaryAttributesAtCharacterIndex:effectiveRange:` (page 1517)
- `temporaryAttribute:atCharacterIndex:effectiveRange:` (page 1515)

Declared In
`NSLayoutManager.h`

temporaryAttributesAtCharacterIndex:effectiveRange:

Returns the dictionary of temporary attributes for the character range specified in *effectiveCharRange* at character index *charIndex*.

```
- (NSDictionary *)temporaryAttributesAtCharacterIndex:(NSUInteger)charIndex
    effectiveRange:(NSRangePointer)effectiveCharRange
```

Return Value
The dictionary of temporary attributes for the character range specified in *effectiveCharRange* at character index *charIndex*.

Discussion
Temporary attributes are used only for onscreen drawing and are not persistent in any way. `NSTextView` uses them to color misspelled words when continuous spell checking is enabled. Currently the only temporary attributes recognized are those that do not affect layout (colors, underlines, and so on).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addTemporaryAttributes:forCharacterRange:` (page 1449)
- `removeTemporaryAttribute:forCharacterRange:` (page 1494)
- `setTemporaryAttributes:forCharacterRange:` (page 1508)

Declared In
`NSLayoutManager.h`

temporaryAttributesAtCharacterIndex:longestEffectiveRange:inRange:

Returns the temporary attributes for the character at a given index, and by reference the maximum range over which the attributes apply.

```
- (NSDictionary *)temporaryAttributesAtCharacterIndex:(NSUInteger)location
    longestEffectiveRange:(NSRangePointer)range inRange:(NSRange)rangeLimit
```

Parameters

location
The index for which to return attributes. This value must not exceed the bounds of the receiver.

range
If not `NULL`, on output, contains the maximum range over which the attributes and values are the same as those at *location*, clipped to *rangeLimit*.

rangeLimit
The range over which to search for continuous presence of the attributes at *location*. This value must not exceed the bounds of the receiver.

Return Value
The attributes for the character at *location*.

Discussion
If you don't need the longest effective range, it's far more efficient to use the `temporaryAttributesAtCharacterIndex:effectiveRange:` (page 1517) method to retrieve the attribute value.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `temporaryAttributesAtCharacterIndex:effectiveRange:` (page 1517)
- `temporaryAttribute:atCharacterIndex:longestEffectiveRange:inRange:` (page 1516)

Declared In
`NSLayoutManager.h`

textContainerChangedGeometry:

Invalidates the layout information, and possibly glyphs, for the given text container and all subsequent `NSTextContainer` objects.

```
- (void)textContainerChangedGeometry:(NSTextContainer *)aTextContainer
```

Parameters
aTextContainer
The text container whose layout is invalidated.

Discussion
This method is invoked automatically by other components of the text system; you should rarely need to invoke it directly. Subclasses of `NSTextContainer`, however, must invoke this method any time their size of shape changes (a text container that dynamically adjusts its shape to wrap text around placed graphics, for example, must do so when a graphic is added, moved, or removed).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSLayoutManager.h`

textContainerChangedTextView:

Updates information needed to manage `NSTextView` objects in the given text container.

```
- (void)textContainerChangedTextView:(NSTextContainer *)aTextContainer
```

Parameters
aTextContainer
The text container whose text view has changed.

Discussion
This method is called by a text container, whenever its text view changes, to keep notifications synchronized. You should rarely need to invoke it directly.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSLayoutManager.h`

textContainerForGlyphAtIndex:effectiveRange:

Returns the container in which the given glyph is laid out and (optionally) by reference the whole range of glyphs that are in that container.

```
- (NSTextContainer *)textContainerForGlyphAtIndex:(NSUInteger)glyphIndex
    effectiveRange:(NSRangePointer)effectiveGlyphRange
```

Parameters

glyphIndex

Index of a glyph in the returned container.

effectiveGlyphRange

If not NULL, on output, points to the whole range of glyphs that are in the returned container.

Return Value

The text container in which the glyph at *glyphIndex* is laid out.

Discussion

This method causes glyph generation and layout for the line fragment containing the specified glyph, or if noncontiguous layout is not enabled, up to and including that line fragment. If noncontiguous layout is not enabled and *effectiveGlyphRange* is not NULL, this method additionally causes glyph generation and layout for the entire text container containing the specified glyph.

Overriding this method is not recommended. Any changes to the returned glyph range should be done at the typesetter level.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setTextContainer:forGlyphRange: (page 1509)

Declared In

NSLayoutManager.h

textContainerForGlyphAtIndex:effectiveRange:withoutAdditionalLayout:

Returns the container in which the given glyph is laid out and (optionally) by reference the whole range of glyphs that are in that container.

```
- (NSTextContainer *)textContainerForGlyphAtIndex:(NSUInteger)glyphIndex
    effectiveRange:(NSRangePointer)effectiveGlyphRange
    withoutAdditionalLayout:(BOOL)flag
```

Parameters

glyphIndex

Index of a glyph in the returned container.

effectiveGlyphRange

If not NULL, on output, points to the whole range of glyphs that are in the returned container.

flag

If YES, glyph generation and layout are not performed, so this option should not be used unless layout is known to be complete for the range in question, or unless noncontiguous layout is enabled; if NO, both are performed as needed.

Return Value
The text container in which the glyph at *glyphIndex* is laid out.

Discussion
This method is primarily for use from within `NSTypesetter`, after layout is complete for the range in question, but before the layout manager's call to `NSTypesetter` has returned. In that case glyph and layout holes have not yet been recalculated, so the layout manager does not yet know that layout is complete for that range, and this variant must be used.

Overriding this method is not recommended. Any changes to the returned glyph range should be done at the typesetter level.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTextContainer:forGlyphRange:` (page 1509)

Declared In
`NSLayoutManager.h`

textContainers

Returns the receiver's text containers.

```
- (NSArray *)textContainers
```

Return Value
The receiver's text containers.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addTextContainer:` (page 1449)
- `insertTextContainer:atIndex:` (page 1480)
- `removeTextContainerAtIndex:` (page 1495)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSLayoutManager.h`

textStorage

Returns the receiver's text storage object.

```
- (NSTextStorage *)textStorage
```

Return Value
The receiver's text storage.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTextStorage:` (page 1509)
- `replaceTextStorage:` (page 1496)

Declared In
`NSLayoutManager.h`

textStorage:edited:range:changeInLength:invalidatedRange:

Invalidates glyph and layout information for a portion of the text in the given text storage object.

```
- (void)textStorage:(NSTextStorage *)aTextStorage edited:(NSUInteger)mask
    range:(NSRange)newCharRange changeInLength:(NSInteger)delta
    invalidatedRange:(NSRange)invalidatedCharRange
```

Parameters

aTextStorage
The text storage whose information is invalidated.

mask
Specifies the nature of the changes. Its value is made by combining with the C bitwise OR operator the constants described in "Change notifications" in `NSTextStorage` (`NSTextStorageEditedAttributes` (page 2848) and `NSTextStorageEditedCharacters` (page 2848)).

newCharRange
Indicates the extent of characters resulting from the edits.

delta
If the `NSTextStorageEditedCharacters` bit of *mask* is set, gives the number of characters added to or removed from the original range (otherwise its value is irrelevant).

invalidatedCharRange
Represents the range of characters affected after attributes have been fixed. Is either equal to *newCharRange* or larger. For example, deleting a paragraph separator character invalidates the layout information for all characters in the paragraphs that precede and follow the separator.

Discussion
This message is sent from the `NSTextStorage`object's `processEditing` (page 2844) method to indicate that its characters or attributes have changed. This method invalidates glyphs and layout for the affected characters.

For example, after replacing "The" with "Several" to produce the string "Several files couldn't be saved", *newCharRange* is {0, 7} and *delta* is 4. The receiver uses this information to update its character-to-glyph mapping and to update the selection range based on the change.

The `textStorage:edited:range:changeInLength:invalidatedRange:` messages are sent in a series to each `NSLayoutManager` object associated with the text storage object, so the layout managers receiving them shouldn't edit *aTextStorage* while this method is executing. If one of them does, the *newCharRange*, *delta*, and *invalidatedCharRange* arguments are incorrect for all following layout managers that receive the message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `invalidateLayoutForCharacterRange:isSoft:actualCharacterRange:` (page 1483)

Declared In
`NSLayoutManager.h`

textViewForBeginningOfSelection

Returns the text view containing the first glyph in the selection.

```
- (NSTextView *)textViewForBeginningOfSelection
```

Return Value
The text view containing the first glyph in the selection, or `nil` if there's no selection or there isn't enough layout information to determine the text view.

Discussion
This method does not cause layout if the beginning of the selected range is not yet laid out.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSLayoutManager.h`

typesetter

Returns the receiver's typesetter.

```
- (NSTypesetter *)typesetter
```

Return Value
The receiver's typesetter.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTypesetter:` (page 1510)

Declared In
`NSLayoutManager.h`

typesetterBehavior

Returns the current typesetter behavior.

```
- (NSTypesetterBehavior)typesetterBehavior
```

Return Value
The current typesetter behavior value.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `setTypesetterBehavior:` (page 1510)

Declared In
`NSLayoutManager.h`

underlineGlyphRange:underlineType:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:

Calculates subranges to be underlined for the glyphs in a given range and draws the underlining as appropriate.

```
- (void)underlineGlyphRange:(NSRange)glyphRange underlineType:(NSInteger)underlineVal
    lineFragmentRect:(NSRect)lineRect lineFragmentGlyphRange:(NSRange)lineGlyphRange
    containerOrigin:(NSPoint)containerOrigin
```

Parameters

glyphRange
A range of glyphs, which must belong to a single line fragment rectangle (as returned by `lineFragmentRectForGlyphAtIndex:effectiveRange:` (page 1487)).

underlineVal
The style of underlining to draw. This value is a mask derived from the value for `NSUnderlineStyleAttributeName` (page 272)—for example, `(NSUnderlinePatternDash | NSUnderlineStyleThick | NSUnderlineByWordMask)`. Subclasses can define custom underlining styles.

lineRect
The line fragment rectangle containing the glyphs to draw underlining for.

lineGlyphRange
The range of all glyphs within that line fragment rectangle.

containerOrigin
The origin of the line fragment rectangle's `NSTextContainer` in its `NSTextView`.

Discussion
This method determines which glyphs actually need to be underlined based on *underlineVal*. With `NSUnderlineStyleSingle`, for example, leading and trailing whitespace isn't underlined, but whitespace between visible glyphs is. A potential word-underline style would omit underlining on any whitespace. After determining which glyphs to draw underlining on, this method invokes `drawUnderlineForGlyphRange:underlineType:baselineOffset:lineFragmentRect:lineFragmentGlyphRange:containerOrigin:` (page 1461) for each contiguous range of glyphs that requires it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textContainerForGlyphAtIndex:effectiveRange:` (page 1519)
- `textContainerOrigin` (page 2956) (`NSTextView`)

Declared In
NSLayoutManager.h

usedRectForTextContainer:

Returns the bounding rectangle for the glyphs laid out in the given text container.

```
- (NSRect)usedRectForTextContainer:(NSTextContainer *)aTextContainer
```

Discussion
Returns the text container's currently used area, which determines the size that the view would need to be in order to display all the glyphs that are currently laid out in the container. This causes neither glyph generation nor layout.

Used rectangles are always in container coordinates.

Availability
Available in Mac OS X v10.0 and later.

See Also
- containerSize (page 2781) (NSTextContainer)

Related Sample Code
CircleView
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
NSLayoutManager.h

usesFontLeading

Indicates whether the receiver uses the leading provided in the font.

```
- (BOOL)usesFontLeading
```

Return Value
YES if the receiver uses the font's leading; otherwise, NO.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setUsesFontLeading: (page 1511)

Declared In
NSLayoutManager.h

usesScreenFonts

Indicates whether the receiver uses screen fonts to calculate layout and display text.

```
- (BOOL)usesScreenFonts
```

Return Value
`YES` if the receiver calculates layout and displays text using screen fonts when possible; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setUsesScreenFonts:` (page 1511)
- `substituteFontForFont:` (page 1514)

Declared In
`NSLayoutManager.h`

Constants

Glyph Attributes

These glyph attribute constants are used only inside the glyph generation machinery, but they must be shared between components.

```
enum {
    NSGlyphAttributeSoft        = 0,
    NSGlyphAttributeElastic     = 1,
    NSGlyphAttributeBidiLevel   = 2,
    NSGlyphAttributeInscribe    = 5
};
```

Constants

`NSGlyphAttributeSoft`
The glyph is soft.
Available in Mac OS X v10.0 and later.
Declared in `NSLayoutManager.h`.

`NSGlyphAttributeElastic`
The glyph is elastic.
Available in Mac OS X v10.0 and later.
Declared in `NSLayoutManager.h`.

`NSGlyphAttributeBidiLevel`
The bidirectional level (direction) of the glyph.
Available in Mac OS X v10.2 and later.
Declared in `NSLayoutManager.h`.

`NSGlyphAttributeInscribe`
Glyph inscription attribute. See [NSGlyphInscription] for possible values.`NSGlyphInscription` (page 1526)

Available in Mac OS X v10.0 and later.

Declared in `NSLayoutManager.h`.

Declared In
`NSLayoutManager.h`

NSGlyphInscription

These constants specify how a glyph is laid out relative to the previous glyph. The glyph inscription constants are possible values for the glyph attribute `NSGlyphAttributeInscribe`. Glyph inscriptions are set during glyph generation.

```
typedef enum {
    NSGlyphInscribeBase = 0,
    NSGlyphInscribeBelow = 1,
    NSGlyphInscribeAbove = 2,
    NSGlyphInscribeOverstrike = 3,
    NSGlyphInscribeOverBelow = 4
} NSGlyphInscription;
```

Constants
`NSGlyphInscribeBase`
A base glyph; a character that the font can represent with a single glyph.

Available in Mac OS X v10.0 and later.

Declared in `NSLayoutManager.h`.

`NSGlyphInscribeBelow`
Glyph is rendered below the previous glyph.

Available in Mac OS X v10.0 and later.

Declared in `NSLayoutManager.h`.

`NSGlyphInscribeAbove`
Glyph is rendered above the previous glyph.

Available in Mac OS X v10.0 and later.

Declared in `NSLayoutManager.h`.

`NSGlyphInscribeOverstrike`
Glyph is rendered on top of the previous glyph.

Available in Mac OS X v10.0 and later.

Declared in `NSLayoutManager.h`.

`NSGlyphInscribeOverBelow`
Glyph is rendered on top and below the previous glyph.

Available in Mac OS X v10.0 and later.

Declared in `NSLayoutManager.h`.

Discussion
The only constants that the text system currently uses are `NSGlyphInscribeBase` (for most glyphs) and `NSGlyphInscribeOverstrike` (for nonbase glyphs). Nonbase glyphs occur when diacritical marks are applied to a base character, and the font does not have a single glyph to represent the combination. For

example, if a font did not contain a single glyph for ü, but did contain separate glyphs for u and ¨, then it could be rendered with a base glyph u followed by a nonbase glyph ¨. In that case the nonbase glyph would have the value `NSGlyphInscribeOverstrike` for the inscribe attribute.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSLayoutManager.h`

NSTypesetterBehavior

These constants define the behavior of `NSLayoutManager` and `NSTypesetter` when laying out lines. They are used by `setTypesetterBehavior:` (page 1510) and `typesetterBehavior` (page 1522) to control the compatibility level of the typesetter.

```
typedef enum {
    NSTypesetterLatestBehavior = -1,
    NSTypesetterOriginalBehavior = 0,
    NSTypesetterBehavior_10_2_WithCompatibility = 1,
    NSTypesetterBehavior_10_2 = 2,
    NSTypesetterBehavior_10_3 = 3,
    NSTypesetterBehavior_10_4 = 4
} NSTypesetterBehavior;
```

Constants

`NSTypesetterLatestBehavior`
The most current typesetter behavior in the current system version. For Mac OS X v10.2, this behavior is identical to `NSTypesetterBehavior_10_2`. If you use this behavior setting, you cannot necessarily rely on line width and height metrics remaining the same across different versions of Mac OS X.

Available in Mac OS X v10.2 and later.

Declared in `NSLayoutManager.h`.

`NSTypesetterOriginalBehavior`
The original typesetter behavior, as shipped with Mac OS X v10.1 and earlier.

Available in Mac OS X v10.2 and later.

Declared in `NSLayoutManager.h`.

`NSTypesetterBehavior_10_2_WithCompatibility`
Typesetting same as `NSTypesetterBehavior_10_2` but using line widths and height metric calculations that are the same as with `NSTypesetterOriginalBehavior`.

Available in Mac OS X v10.2 and later.

Declared in `NSLayoutManager.h`.

`NSTypesetterBehavior_10_2`
The typesetter behavior introduced in Mac OS X version 10.2. This typesetter behavior provides enhanced line and character spacing accuracy and supports more languages than the original typesetter behavior.

Available in Mac OS X v10.2 and later.

Declared in `NSLayoutManager.h`.

`NSTypesetterBehavior_10_3`
: The typesetter behavior introduced in Mac OS X version 10.3.

  Available in Mac OS X v10.3 and later.

  Declared in `NSLayoutManager.h`.

`NSTypesetterBehavior_10_4`
: The typesetter behavior introduced in Mac OS X version 10.4.

  Available in Mac OS X v10.4 and later.

  Declared in `NSLayoutManager.h`.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSLayoutManager.h`

NSLevelIndicator Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSLevelIndicator.h
Availability	Available in Mac OS X v10.4 and later.
Related sample code	ButtonMadness From A View to A Movie QTRecorder

Overview

NSLevelIndicator is a subclass of NSControl that displays a value on a linear scale.

Level indicators provide a visual representation of a level or amount of something, using discrete values. While similar to NSSlider, it provides a more customized visual feedback to the user. Level indicators do not have a "knob" indicating the current setting or allowing the user to adjust settings. The supported indicator styles include:

- A capacity style level indicator. The continuous mode for this style is often used to indicate conditions such as how much data is on hard disk. The discrete mode is similar to audio level indicators in audio playback applications. You can specify both a warning value and a critical value that provides additional visual feedback to the user.
- A ranking style level indicator. This is similar to the star ranking displays provided in iTunes and iPhoto. You can also specify your own ranking image.
- A relevancy style level indicator. This style is used to display the relevancy of a search result, for example in Mail.

NSLevelIndicator uses an NSLevelIndicatorCell to implement much of the control's functionality. NSLevelIndicator provides cover methods for most of NSLevelIndicatorCell's methods, which invoke the corresponding cell method.

Tasks

Configuring the Range of Values

- `setMinValue:` (page 1533)
  Sets the minimum value the receiver can represent to *`minValue`*.
- `minValue` (page 1531)
  Returns the receiver's minimum value.
- `setMaxValue:` (page 1533)
  Sets the maximum value the receiver can represent to *`maxValue`*.
- `maxValue` (page 1531)
  Returns the receiver's maximum value.
- `setWarningValue:` (page 1535)
  Sets the receiver's warning value to *`warningValue`*.
- `warningValue` (page 1536)
  Returns the receiver's warning value.
- `setCriticalValue:` (page 1532)
  Sets the receiver's critical value to *`criticalValue`*.
- `criticalValue` (page 1531)
  Returns the receiver's critical value.

Managing Tick Marks

- `setTickMarkPosition:` (page 1534)
  Sets where tick marks appear relative to the receiver.
- `tickMarkPosition` (page 1535)
  Returns how the receiver's tick marks are aligned with it.
- `setNumberOfTickMarks:` (page 1534)
  Sets the number of tick marks displayed by the receiver (which include those assigned to the minimum and maximum values) to *`count`*.
- `numberOfTickMarks` (page 1532)
  Returns the number of tick marks associated with the receiver.
- `setNumberOfMajorTickMarks:` (page 1533)
  Sets the number of major tick marks displayed by the receiver.
- `numberOfMajorTickMarks` (page 1531)
  Returns the number of major tick marks associated with the receiver.
- `tickMarkValueAtIndex:` (page 1535)
  Returns the receiver's value represented by the tick mark at index (the minimum-value tick mark has an index of 0).
- `rectOfTickMarkAtIndex:` (page 1532)
  Returns the bounding rectangle of the tick mark identified by *`index`* (the minimum-value tick mark is at index 0).

Instance Methods

criticalValue

Returns the receiver's critical value.

```
- (double)criticalValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setCriticalValue: (page 1532)

Declared In
NSLevelIndicator.h

maxValue

Returns the receiver's maximum value.

```
- (double)maxValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setMaxValue: (page 1533)

Declared In
NSLevelIndicator.h

minValue

Returns the receiver's minimum value.

```
- (double)minValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setMinValue: (page 1533)

Declared In
NSLevelIndicator.h

numberOfMajorTickMarks

Returns the number of major tick marks associated with the receiver.

```
- (NSInteger)numberOfMajorTickMarks
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setNumberOfMajorTickMarks: (page 1533)

Declared In
NSLevelIndicator.h

numberOfTickMarks

Returns the number of tick marks associated with the receiver.

```
- (NSInteger)numberOfTickMarks
```

Discussion
The tick marks assigned to the minimum and maximum values are included.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setNumberOfTickMarks: (page 1534)

Declared In
NSLevelIndicator.h

rectOfTickMarkAtIndex:

Returns the bounding rectangle of the tick mark identified by *index* (the minimum-value tick mark is at index 0).

```
- (NSRect)rectOfTickMarkAtIndex:(NSInteger)index
```

Discussion
If no tick mark is associated with *index*, the method raises a NSRangeException.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSLevelIndicator.h

setCriticalValue:

Sets the receiver's critical value to *criticalValue*.

```
- (void)setCriticalValue:(double)criticalValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `criticalValue` (page 1531)

Declared In
`NSLevelIndicator.h`

setMaxValue:

Sets the maximum value the receiver can represent to *maxValue*.

```
- (void)setMaxValue:(double)maxValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `maxValue` (page 1531)

Declared In
`NSLevelIndicator.h`

setMinValue:

Sets the minimum value the receiver can represent to *minValue*.

```
- (void)setMinValue:(double)minValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `minValue` (page 1531)

Declared In
`NSLevelIndicator.h`

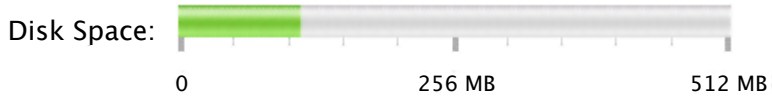
setNumberOfMajorTickMarks:

Sets the number of major tick marks displayed by the receiver.

```
- (void)setNumberOfMajorTickMarks:(NSInteger)count
```

Discussion
The *count* must be less than or equal to the number of tick marks returned by `numberOfTickMarks` (page 1532). For example, if the number of tick marks is 11 and you specify 3 major tick marks, the resulting level indicator will display 3 major tickmarks alternating with 8 minor tick marks, as in the example shown in Figure 66-1.

Figure 66-1 Major and minor tick marks in a level indicator



Availability
Available in Mac OS X v10.4 and later.

See Also
- `numberOfMajorTickMarks` (page 1531)

Declared In
`NSLevelIndicator.h`

setNumberOfTickMarks:

Sets the number of tick marks displayed by the receiver (which include those assigned to the minimum and maximum values) to *count*.

```
- (void)setNumberOfTickMarks:(NSInteger)count
```

Discussion
By default, this value is 0, and no tick marks appear. The number of tick marks assigned to a slider, along with the slider's minimum and maximum values, determines the values associated with the tick marks.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `numberOfTickMarks` (page 1532)

Declared In
`NSLevelIndicator.h`

setTickMarkPosition:

Sets where tick marks appear relative to the receiver.

```
- (void)setTickMarkPosition:(NSTickMarkPosition)position
```

Discussion
This method has no effect if no tick marks have been assigned (that is, `numberOfTickMarks` (page 1532) returns 0).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `tickMarkPosition` (page 1535)

Declared In
NSLevelIndicator.h

setWarningValue:

Sets the receiver's warning value to *warningValue*.

```
- (void)setWarningValue:(double)warningValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- warningValue (page 1536)

Declared In
NSLevelIndicator.h

tickMarkPosition

Returns how the receiver's tick marks are aligned with it.

```
- (NSTickMarkPosition)tickMarkPosition
```

Discussion
The default alignments are NSTickMarkBelow and NSTickMarkLeft.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setTickMarkPosition: (page 1534)

Declared In
NSLevelIndicator.h

tickMarkValueAtIndex:

Returns the receiver's value represented by the tick mark at index (the minimum-value tick mark has an index of 0).

```
- (double)tickMarkValueAtIndex:(NSInteger)index
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSLevelIndicator.h

warningValue

Returns the receiver's warning value.

```
- (double)warningValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setWarningValue:` (page 1535)

Declared In
`NSLevelIndicator.h`

NSLevelIndicatorCell Class Reference

Inherits from	NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSLevelIndicatorCell.h
Availability	Available in Mac OS X v10.4 and later.

Overview

`NSLevelIndicatorCell` is a subclass of `NSActionCell` that provides several level indicator display styles including: capacity, ranking and relevancy. The capacity style provides both continuous and discrete modes.

Tasks

Initializing NSLevelIndicatorCell Objects

- `initWithLevelIndicatorStyle:` (page 1539)
  Initializes the receiver with the style specified by *`levelIndicatorStyle`*.

Configuring the Range of Values

- `setMinValue:` (page 1542)
  Sets the minimum value the receiver can represent to *`minValue`*.
- `minValue` (page 1540)
  Returns the receiver's minimum value.
- `setMaxValue:` (page 1542)
  Sets the maximum value the receiver can represent to *`maxValue`*.
- `maxValue` (page 1540)
  Returns the receiver's maximum value.
- `setLevelIndicatorStyle:` (page 1542)
  Sets the style of the receiver to *`levelIndicatorStyle`*.

- `levelIndicatorStyle` (page 1539)
  Returns the level indicator style of the receiver.
- `setWarningValue:` (page 1544)
  Sets the receiver's warning value to *warningValue*.
- `warningValue` (page 1545)
  Returns the receiver's warning value.
- `setCriticalValue:` (page 1541)
  Sets the receiver's critical value to *criticalValue*.
- `criticalValue` (page 1539)
  Returns the receiver's critical value.

Managing Tick Marks

- `setTickMarkPosition:` (page 1543)
  Sets where tick marks appear relative to the receiver.
- `tickMarkPosition` (page 1544)
  Returns how the receiver's tick marks are aligned with it.
- `setNumberOfTickMarks:` (page 1543)
  Sets the number of tick marks displayed by the receiver (which include those assigned to the minimum and maximum values) to *numberOfTickMarks*.
- `numberOfTickMarks` (page 1540)
  Returns the number of tick marks associated with the receiver.
- `setNumberOfMajorTickMarks:` (page 1543)
  Sets the number of major tick marks displayed by the receiver.
- `numberOfMajorTickMarks` (page 1540)
  Returns the number of major tick marks associated with the receiver.
- `tickMarkValueAtIndex:` (page 1544)
  Returns the receiver's value represented by the tick mark at index (the minimum-value tick mark has an index of 0).
- `rectOfTickMarkAtIndex:` (page 1541)
  Returns the bounding rectangle of the tick mark identified by *index* (the minimum-value tick mark is at index 0).

Setting the Level-Indicator Image

- `setImage:` (page 1541)
  Sets the image displayed by the receiver for the `NSRatingLevelIndicatorStyle` to *image*.

Instance Methods

criticalValue

Returns the receiver's critical value.

```
- (double)criticalValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setCriticalValue: (page 1541)

Declared In
NSLevelIndicatorCell.h

initWithLevelIndicatorStyle:

Initializes the receiver with the style specified by *levelIndicatorStyle*.

```
- (id)initWithLevelIndicatorStyle:(NSLevelIndicatorStyle)levelIndicatorStyle
```

Discussion
The default value and minimum value are 0.0. The default maximum value is dependent on *levelIndicatorStyle*. For continuous styles, the default maximum value is 100.0. For discrete styles the default maximum value is 5.0.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSLevelIndicatorCell.h

levelIndicatorStyle

Returns the level indicator style of the receiver.

```
- (NSLevelIndicatorStyle)levelIndicatorStyle
```

Discussion
Possible return values are described in "Constants" (page 1545).

Availability
Available in Mac OS X v10.4 and later.

See Also
- setLevelIndicatorStyle: (page 1542)

Declared In
NSLevelIndicatorCell.h

maxValue

Returns the receiver's maximum value.

```
- (double)maxValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setLevelIndicatorStyle:` (page 1542)

Declared In
`NSLevelIndicatorCell.h`

minValue

Returns the receiver's minimum value.

```
- (double)minValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setMinValue:` (page 1542)

Declared In
`NSLevelIndicatorCell.h`

numberOfMajorTickMarks

Returns the number of major tick marks associated with the receiver.

```
- (NSInteger)numberOfMajorTickMarks
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setNumberOfMajorTickMarks:` (page 1543)

Declared In
`NSLevelIndicatorCell.h`

numberOfTickMarks

Returns the number of tick marks associated with the receiver.

```
- (NSInteger)numberOfTickMarks
```

Discussion
The tick marks assigned to the minimum and maximum values are included.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setNumberOfTickMarks:` (page 1543)

Declared In
`NSLevelIndicatorCell.h`

rectOfTickMarkAtIndex:

Returns the bounding rectangle of the tick mark identified by *index* (the minimum-value tick mark is at index 0).

```
- (NSRect)rectOfTickMarkAtIndex:(NSInteger)index
```

Discussion
If no tick mark is associated with *index*, the method raises a `NSRangeException`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSLevelIndicatorCell.h`

setCriticalValue:

Sets the receiver's critical value to *criticalValue*.

```
- (void)setCriticalValue:(double)criticalValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `criticalValue` (page 1539)

Declared In
`NSLevelIndicatorCell.h`

setImage:

Sets the image displayed by the receiver for the `NSRatingLevelIndicatorStyle` to *image*.

```
- (void)setImage:(NSImage *)image
```

Discussion
The image is lightened to indicate a highlighted selection and dots are drawn for empty spots. The image is not stretched and no space is added between images. Setting *image* to `nil` causes the default star image to be used.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSLevelIndicatorCell.h

setLevelIndicatorStyle:

Sets the style of the receiver to *levelIndicatorStyle*.

```
- (void)setLevelIndicatorStyle:(NSLevelIndicatorStyle)levelIndicatorStyle
```

Discussion
The available values of *levelIndicatorStyle* are described in "Constants" (page 1545).

Availability
Available in Mac OS X v10.4 and later.

See Also
- levelIndicatorStyle (page 1539)

Related Sample Code
ButtonMadness

Declared In
NSLevelIndicatorCell.h

setMaxValue:

Sets the maximum value the receiver can represent to *maxValue*.

```
- (void)setMaxValue:(double)maxValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- levelIndicatorStyle (page 1539)

Declared In
NSLevelIndicatorCell.h

setMinValue:

Sets the minimum value the receiver can represent to *minValue*.

```
- (void)setMinValue:(double)minValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `minValue` (page 1540)

Declared In
`NSLevelIndicatorCell.h`

setNumberOfMajorTickMarks:

Sets the number of major tick marks displayed by the receiver.

```
- (void)setNumberOfMajorTickMarks:(NSInteger)count
```

Discussion
The *count* must be less than or equal to the number of tick marks.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `numberOfMajorTickMarks` (page 1540)

Declared In
`NSLevelIndicatorCell.h`

setNumberOfTickMarks:

Sets the number of tick marks displayed by the receiver (which include those assigned to the minimum and maximum values) to *numberOfTickMarks*.

```
- (void)setNumberOfTickMarks:(NSInteger)count
```

Discussion
By default, this value is 0, and no tick marks appear. The number of tick marks assigned to a slider, along with the slider's minimum and maximum values, determines the values associated with the tick marks.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `numberOfTickMarks` (page 1540)

Declared In
`NSLevelIndicatorCell.h`

setTickMarkPosition:

Sets where tick marks appear relative to the receiver.

```
- (void)setTickMarkPosition:(NSTickMarkPosition)position
```

Discussion
This method has no effect if no tick marks have been assigned (that is, `numberOfTickMarks` (page 1540) returns 0).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- tickMarkPosition` (page 1544)

Declared In
`NSLevelIndicatorCell.h`

setWarningValue:

Sets the receiver's warning value to *warningValue*.

```
- (void)setWarningValue:(double)warningValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`- warningValue` (page 1545)

Declared In
`NSLevelIndicatorCell.h`

tickMarkPosition

Returns how the receiver's tick marks are aligned with it.

```
- (NSTickMarkPosition)tickMarkPosition
```

Discussion
The default alignments are `NSTickMarkBelow` and `NSTickMarkLeft`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setTickMarkPosition:` (page 1543)

Declared In
`NSLevelIndicatorCell.h`

tickMarkValueAtIndex:

Returns the receiver's value represented by the tick mark at index (the minimum-value tick mark has an index of 0).

```
- (double)tickMarkValueAtIndex:(NSInteger)index
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSLevelIndicatorCell.h

warningValue

Returns the receiver's warning value.

```
- (double)warningValue
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setWarningValue: (page 1544)

Declared In
NSLevelIndicatorCell.h

Constants

NSLevelIndicatorStyle

The following constants specify the level indicator's style and are used by initWithLevelIndicatorStyle: (page 1539), levelIndicatorStyle (page 1539), and setLevelIndicatorStyle: (page 1542).

```
enum {
    NSRelevancyLevelIndicatorStyle,
    NSContinuousCapacityLevelIndicatorStyle,
    NSDiscreteCapacityLevelIndicatorStyle,
    NSRatingLevelIndicatorStyle
};
typedef NSUInteger NSLevelIndicatorStyle;
```

Constants

NSRelevancyLevelIndicatorStyle
A style similar to the rank column displayed when searching in Mail.app.
Available in Mac OS X v10.4 and later.
Declared in NSLevelIndicatorCell.h.

NSRatingLevelIndicatorStyle
A style similar to the star ranking displays provided in iTunes and iPhoto.
Available in Mac OS X v10.4 and later.
Declared in NSLevelIndicatorCell.h.

`NSDiscreteCapacityLevelIndicatorStyle`
A style similar to audio level indicators in iTunes.
Available in Mac OS X v10.4 and later.
Declared in `NSLevelIndicatorCell.h`.

`NSContinuousCapacityLevelIndicatorStyle`
A style that is often used to indicate conditions such as how much data is on a hard disk.
Available in Mac OS X v10.4 and later.
Declared in `NSLevelIndicatorCell.h`.

NSMatrix Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSMatrix.h
Companion guide	Matrix Programming Guide
Related sample code	ButtonMadness DatePicker EnhancedDataBurn Quartz Composer WWDC 2005 TextEdit Sketch-112

Overview

`NSMatrix` is a class used for creating groups of `NSCell` objects that work together in various ways.

The cells in an `NSMatrix` object are numbered by row and column, each starting with 0; for example, the top left `NSCell` would be at (0, 0), and the `NSCell` that's second down and third across would be at (1, 2). The `NSMatrix` class has the notion of a single selected cell, which is the cell that was most recently clicked or that was so designated by a `selectCellAtRow:column:` (page 1578) or `selectCellWithTag:` (page 1579) message. The selected cell is the cell chosen for action messages except for `performClick:` (page 573) (`NSCell`), which is assigned to the key cell. (The key cell is generally identical to the selected cell, but can be given click focus while leaving the selected cell unchanged.) If the user has selected multiple cells, the selected cell is the one lowest and furthest to the right in the matrix of cells.

Tasks

Initializing an NSMatrix Object

- `initWithFrame:` (page 1565)
  Initializes a newly allocated matrix with the specified frame.
- `initWithFrame:mode:cellClass:numberOfRows:numberOfColumns:` (page 1566)
  Initializes and returns a newly allocated matrix of the specified size using cells of the given class.
- `initWithFrame:mode:prototype:numberOfRows:numberOfColumns:` (page 1566)
  Initializes and returns a newly allocated matrix of the specified size using the given cell as a prototype.

Configuring the Matrix Object

- `setMode:` (page 1589)
  Sets the selection mode of the receiver.
- `mode` (page 1572)
  Returns the selection mode of the matrix.
- `setAllowsEmptySelection:` (page 1583)
  Sets whether a radio-mode matrix allows an empty selection.
- `allowsEmptySelection` (page 1556)
  Returns a Boolean value indicating whether a radio-mode matrix supports an empty selection.
- `setSelectionByRect:` (page 1590)
  Sets whether the user can select a rectangle of cells in the receiver by dragging the cursor.
- `isSelectionByRect` (page 1570)
  Returns a Boolean value indicating whether the user can drag the cursor to select a rectangle of cells in the matrix.

Managing the Cell Class

- `setCellClass:` (page 1585)
  Configures the receiver to use instances of the specified class when creating new cells.
- `cellClass` (page 1558)
  Returns the class that the matrix uses to create new cells.
- `setPrototype:` (page 1589)
  Sets the prototype cell that's copied whenever the matrix creates a new cell.
- `prototype` (page 1574)
  Returns the prototype cell that's copied when a new cell is created.,

Laying Out the Cells of the Matrix

- `addColumn` (page 1554)
  Adds a new column of cells to the right of the last column.

- `addColumnWithCells:` (page 1554)
  Adds a new column of cells to the right of the last column, using the given cells.
- `addRow` (page 1555)
  Adds a new row of cells below the last row.
- `addRowWithCells:` (page 1556)
  Adds a new row of cells below the last row, using the specified cells.
- `cellFrameAtRow:column:` (page 1559)
  Returns the frame rectangle of the cell that would be drawn at the specified location.
- `cellSize` (page 1559)
  Returns the size of each cell in the matrix.
- `getNumberOfRows:columns:` (page 1563)
  Obtains the number of rows and columns in the receiver.
- `insertColumn:` (page 1567)
  Inserts a new column of cells at the specified location. .
- `insertColumn:withCells:` (page 1568)
  Inserts a new column of cells before the specified column, using the given cells.
- `insertRow:` (page 1568)
  Inserts a new row of cells before the specified row.
- `insertRow:withCells:` (page 1569)
  Inserts a new row of cells before the specified row, using the given cells.
- `intercellSpacing` (page 1569)
  Returns the spacing between cells in the matrix.
- `makeCellAtRow:column:` (page 1571)
  Creates a new cell at the location specified by the given row and column in the receiver.
- `numberOfColumns` (page 1573)
  Returns the number of columns in the receiver.
- `numberOfRows` (page 1573)
  Returns the number of rows in the receiver.
- `putCell:atRow:column:` (page 1575)
  Replaces the cell at the specified row and column with the new cell.
- `removeColumn:` (page 1575)
  Removes the specified column at from the receiver.
- `removeRow:` (page 1576)
  Removes the specified row from the receiver.
- `renewRows:columns:` (page 1576)
  Changes the number of rows and columns in the receiver.
- `setCellSize:` (page 1586)
  Sets the width and height of each of the cells in the matrix.
- `setIntercellSpacing:` (page 1588)
  Sets the spacing between cells in the matrix.
- `sortUsingFunction:context:` (page 1594)
  Sorts the receiver's cells in ascending order as defined by the specified comparison function.
- `sortUsingSelector:` (page 1594)
  Sorts the receiver's cells in ascending order as defined by the comparison method.

Finding Matrix Coordinates

- `getRow:column:forPoint:` (page 1564)
  Indicates whether the specified point lies within one of the cells of the matrix and returns the location of the cell within which the point lies.
- `getRow:column:ofCell:` (page 1564)
  Searches the receiver for the specified cell and returns the row and column of the cell

Managing Attributes of Individual Cells

- `setState:atRow:column:` (page 1591)
  Sets the state of the cell at specified location.
- `setToolTip:forCell:` (page 1593)
  Sets the tooltip for the cell.
- `toolTipForCell:` (page 1598)
  Returns the tooltip for the specified cell.

Selecting and Deselecting Cells

- `selectCellAtRow:column:` (page 1578)
  Selects the cell at the specified row and column within the receiver.
- `selectCellWithTag:` (page 1579)
  Selects the last cell with the given tag.
- `selectAll:` (page 1578)
  Selects and highlights all cells in the receiver.
- `setKeyCell:` (page 1589)
  Sets the cell that will be clicked when the user presses the Space bar.
- `keyCell` (page 1571)
  Returns the cell that will be clicked when the user presses the Space bar.
- `setSelectionFrom:to:anchor:highlight:` (page 1591)
  Programmatically selects a range of cells.
- `deselectAllCells` (page 1561)
  Deselects all cells in the receiver and, if necessary, redisplays the receiver.
- `deselectSelectedCell` (page 1561)
  Deselects the selected cell or cells.

Finding Cells

- `selectedCell` (page 1579)
  Returns the most recently selected cell.
- `selectedCells` (page 1580)
  Returns the receiver's selected and highlighted cells.

- `selectedColumn` (page 1580)
  Returns the column of the selected cell.
- `selectedRow` (page 1580)
  Returns the row of the selected cell.
- `cellAtRow:column:` (page 1557)
  Returns the cell at the specified row and column.
- `cellWithTag:` (page 1560)
  Searches the receiver and returns the last cell matching the specified tag.
- `cells` (page 1559)
  Returns the cells of the matrix.

Modifying Graphics Attributes

- `backgroundColor` (page 1557)
  Returns the background color of the matrix.
- `cellBackgroundColor` (page 1558)
  Returns the background color of the matrix's cells.
- `drawsBackground` (page 1563)
  Returns a Boolean value indicating whether the matrix draws its background.
- `drawsCellBackground` (page 1563)
  Returns whether the matrix draws the background within each of its cells.
- `setBackgroundColor:` (page 1584)
  Sets the background color for the receiver and redraws the receiver.
- `setCellBackgroundColor:` (page 1585)
  Sets the background color for the cells in the receiver
- `setDrawsBackground:` (page 1587)
  Sets whether the receiver draws its background.
- `setDrawsCellBackground:` (page 1588)
  Sets whether the receiver draws the background within each of its cells.

Editing Text in Cells

- `selectText:` (page 1581)
  Selects text in the currently selected cell or in the key cell.
- `selectTextAtRow:column:` (page 1581)
  Selects the text in the cell at the specified location and returns the cell.
- `textShouldBeginEditing:` (page 1597)
  Requests permission to begin editing text.
- `textDidBeginEditing:` (page 1595)
  Invoked when there's a change in the text after the receiver gains first responder status.
- `textDidChange:` (page 1596)
  Invoked when a key-down event or paste operation occurs that changes the receiver's contents.

- `textShouldEndEditing:` (page 1597)
  Requests permission to end editing.
- `textDidEndEditing:` (page 1596)
  Invoked when text editing ends.

Setting Tab Key Behavior

- `setTabKeyTraversesCells:` (page 1592)
  Sets whether pressing the Tab key advances the key cell to the next selectable cell.
- `tabKeyTraversesCells` (page 1595)
  Returns a Boolean value indicating whether pressing the Tab key advances the key cell to the next selectable cell.

Managing the Delegate

- `delegate` (page 1560)
  Returns the delegate for messages from the field editor.
- `setDelegate:` (page 1586)
  Sets the delegate for messages from the field editor.

Resizing the Matrix and Its Cells

- `setAutosizesCells:` (page 1584)
  Sets whether the cell sizes change when the receiver is resized.
- `autosizesCells` (page 1556)
  Returns a Boolean value indicating whether the matrix automatically resizes it cells.
- `setValidateSize:` (page 1593)
  Specifies whether the receiver's size information is validated.
- `sizeToCells` (page 1593)
  Changes the width and the height of the receiver’s frame so it exactly contains the cells.

Scrolling Cells in the Matrix

- `setAutoscroll:` (page 1584)
  Sets whether the receiver is automatically scrolled.
- `isAutoscroll` (page 1570)
  Returns a Boolean value indicating whether the receiver is automatically scrolled.
- `setScrollable:` (page 1590)
  Specifies whether the cells in the matrix are scrollable.
- `scrollCellToVisibleAtRow:column:` (page 1577)
  Scrolls the receiver so the specified cell is visible.

Displaying and Highlighting Cells

- `drawCellAtRow:column:` (page 1562)
  Displays the cell at the specified row and column.
- `highlightCell:atRow:column:` (page 1565)
  Highlights or unhighlights the cell at the specified row and column location.

Managing and Sending Action Messages

- `sendAction` (page 1582)
  If the selected cell has both an action and a target, sends its action to its target.
- `sendAction:to:forAllCells:` (page 1582)
  Iterates through the cells in the receiver, sending the specified selector to an object for each cell.
- `setDoubleAction:` (page 1587)
  Sets the action sent to the target of the receiver when the user double-clicks a cell.
- `doubleAction` (page 1562)
  Returns the matrix's double-click action method.
- `sendDoubleAction` (page 1583)
  Sends the double-click action message to the target of the receiver.

Handling Event and Action Messages

- `acceptsFirstMouse:` (page 1553)
  Returns a Boolean value indicating whether the receiver accepts the first mouse.
- `mouseDown:` (page 1572)
  Responds to a mouse-down event.
- `mouseDownFlags` (page 1573)
  Returns the flags in effect at the mouse-down event that started the current tracking session.
- `performKeyEquivalent:` (page 1574)
  Looks for a cell that has the given key equivalent and, if found, makes that cell respond as if clicked.

Managing the Cursor

- `resetCursorRects` (page 1577)
  Resets cursor rectangles so the cursor becomes an I-beam over text cells.

Instance Methods

acceptsFirstMouse:

Returns a Boolean value indicating whether the receiver accepts the first mouse.

```
- (BOOL)acceptsFirstMouse:(NSEvent *)theEvent
```

Parameters

theEvent

This parameter is ignored.

Return Value

`NO` if the selection mode of the receiver is `NSListModeMatrix`, `YES` if the receiver is in any other selection mode. The receiver does not accept first mouse in `NSListModeMatrix` to prevent the loss of multiple selections.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `mode` (page 1572)

Declared In

`NSMatrix.h`

addColumn

Adds a new column of cells to the right of the last column.

```
- (void)addColumn
```

Discussion

This method raises an `NSRangeException` if there are 0 rows or 0 columns. This method creates new cells as needed with `makeCellAtRow:column:` (page 1571). Use `renewRows:columns:` (page 1576) to add new cells to an empty matrix.

If the number of rows or columns in the receiver has been changed with `renewRows:columns:` (page 1576), new cells are created only if they are needed. This fact allows you to grow and shrink an `NSMatrix` without repeatedly creating and freeing the cells.

This method redraws the receiver. Your code may need to send `sizeToCells` (page 1593) after sending this method to resize the receiver to fit the newly added cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `cellClass` (page 1558)
- `insertColumn:` (page 1567)
- `prototype` (page 1574)
- `addRow` (page 1555)

Declared In

`NSMatrix.h`

addColumnWithCells:

Adds a new column of cells to the right of the last column, using the given cells.

```
- (void)addColumnWithCells:(NSArray *)newCells
```

Parameters

newCells

An array of objects to use when filling the new column starting with the object at index 0. Each object in should be an instance of `NSCell` or one of its subclasses (usually `NSActionCell`). The array should have a sufficient number of cells to fill the entire column. Extra cells are ignored, unless the matrix is empty. In that case, a matrix is created with one column and enough rows for all the elements of *newCells*.

Discussion

This method redraws the receiver. Your code may need to send `sizeToCells` (page 1593) after sending this method to resize the receiver to fit the newly added cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `insertColumn:withCells:` (page 1568)
- `addRowWithCells:` (page 1556)

Declared In

`NSMatrix.h`

addRow

Adds a new row of cells below the last row.

```
- (void)addRow
```

Discussion

New cells are created as needed with `makeCellAtRow:column:` (page 1571). This method raises an `NSRangeException` if there are 0 rows or 0 columns. Use `renewRows:columns:` (page 1576) to add new cells to an empty matrix.

If the number of rows or columns in the receiver has been changed with `renewRows:columns:` (page 1576), then new cells are created only if they are needed. This fact allows you to grow and shrink an NSMatrix without repeatedly creating and freeing the cells.

This method redraws the receiver. Your code may need to send `sizeToCells` (page 1593) after sending this method to resize the receiver to fit the newly added cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `cellClass` (page 1558)
- `insertRow:` (page 1568)
- `prototype` (page 1574)
- `addColumn` (page 1554)

Declared In

`NSMatrix.h`

addRowWithCells:

Adds a new row of cells below the last row, using the specified cells.

```
- (void)addRowWithCells:(NSArray *)newCells
```

Parameters

newCells

An array of objects to use to fill the new row, starting with the object at index 0. Each object should be an instance of `NSCell` or one of its subclasses (usually `NSActionCell`). The array should contain a sufficient number of cells to fill the entire row. Extra cells are ignored, unless the matrix is empty. In that case, a matrix is created with one row and enough columns for all the elements of *newCells*.

Discussion

This method redraws the receiver. Your code may need to send `sizeToCells` (page 1593) after sending this method to resize the receiver to fit the newly added cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `insertRow:withCells:` (page 1569)
- `addColumnWithCells:` (page 1554)

Declared In

`NSMatrix.h`

allowsEmptySelection

Returns a Boolean value indicating whether a radio-mode matrix supports an empty selection.

```
- (BOOL)allowsEmptySelection
```

Return Value

`YES` if it is possible to have no cells selected in a radio-mode matrix; otherwise `NO`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `mode` (page 1572)
- `setAllowsEmptySelection:` (page 1583)

Declared In

`NSMatrix.h`

autosizesCells

Returns a Boolean value indicating whether the matrix automatically resizes it cells.

```
- (BOOL)autosizesCells
```

Return Value
`YES` if cells are resized proportionally to the receiver when its size changes (and intercell spacing is kept constant). `NO` if the cell size and intercell spacing remain constant.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAutosizesCells:` (page 1584)

Declared In
`NSMatrix.h`

backgroundColor

Returns the background color of the matrix.

```
- (NSColor *)backgroundColor
```

Return Value
The color used to draw the background of the receiver (the space between the cells).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- cellBackgroundColor` (page 1558)
`- drawsBackground` (page 1563)
`- setBackgroundColor:` (page 1584)

Declared In
`NSMatrix.h`

cellAtRow:column:

Returns the cell at the specified row and column.

```
- (id)cellAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters
row
The number of the row containing the cell to return.
column
The number of the column containing the cell to return.

Return Value
The `NSCell` object at the specified row and column location specified, or `nil` if either *row* or *column* is outside the bounds of the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `getRow:column:ofCell:` (page 1564)

Related Sample Code
NewsReader
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSMatrix.h`

cellBackgroundColor

Returns the background color of the matrix's cells.

```
- (NSColor *)cellBackgroundColor
```

Return Value
The color used to fill the background of the receiver's cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `backgroundColor` (page 1557)
- `drawsCellBackground` (page 1563)
- `setCellBackgroundColor:` (page 1585)

Declared In
`NSMatrix.h`

cellClass

Returns the class that the matrix uses to create new cells.

```
- (Class)cellClass
```

Return Value
The subclass of `NSCell` that the receiver uses when creating new (empty) cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `prototype` (page 1574)
- `makeCellAtRow:column:` (page 1571)
- `setCellClass:` (page 1585)

Declared In
`NSMatrix.h`

cellFrameAtRow:column:

Returns the frame rectangle of the cell that would be drawn at the specified location.

```
- (NSRect)cellFrameAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row

The row of the cell.

column

The column of the cell.

Return Value
The frame rectangle of the cell (whether or not the specified cell actually exists).

Availability
Available in Mac OS X v10.0 and later.

See Also
- cellSize (page 1559)

Declared In
NSMatrix.h

cells

Returns the cells of the matrix.

```
- (NSArray *)cells
```

Return Value
An array containing the cells of the receiver.

Discussion
The cells in the array are row-ordered; that is, the first row of cells appears first in the array, followed by the second row, and so forth.

Availability
Available in Mac OS X v10.0 and later.

See Also
- cellAtRow:column: (page 1557)

Related Sample Code
SourceView

Declared In
NSMatrix.h

cellSize

Returns the size of each cell in the matrix.

```
- (NSSize)cellSize
```

Return Value
The width and height of each cell in the receiver (all cells in an `NSMatrix` are the same size).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- cellFrameAtRow:column:` (page 1559)
- `- intercellSpacing` (page 1569)
- `- setCellSize:` (page 1586)

Declared In
`NSMatrix.h`

cellWithTag:

Searches the receiver and returns the last cell matching the specified tag.

```
- (id)cellWithTag:(NSInteger)anInt
```

Parameters
anInt
The tag of the cell to return.

Return Value
The last (when viewing the matrix as a row-ordered array) `NSCell` object that has a tag matching *anInt*, or `nil` if no such cell exists

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- selectCellWithTag:` (page 1579)
- `- setTag:` (page 69) (NSActionCell)

Related Sample Code
EnhancedDataBurn

Declared In
`NSMatrix.h`

delegate

Returns the delegate for messages from the field editor.

```
- (id < NSMatrixDelegate >)delegate
```

Return Value
The delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textShouldBeginEditing:` (page 1597)
- `textShouldEndEditing:` (page 1597)
- `setDelegate:` (page 1586)

Declared In
`NSMatrix.h`

deselectAllCells

Deselects all cells in the receiver and, if necessary, redisplays the receiver.

```
- (void)deselectAllCells
```

Discussion
If the selection mode is `NSRadioModeMatrix` and empty selection is not allowed, this method does nothing.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsEmptySelection` (page 1556)
- `mode` (page 1572)
- `selectAll:` (page 1578)

Declared In
`NSMatrix.h`

deselectSelectedCell

Deselects the selected cell or cells.

```
- (void)deselectSelectedCell
```

Discussion
If the selection mode is `NSRadioModeMatrix` and empty selection is not allowed, or if nothing is currently selected, this method does nothing. This method doesn't redisplay the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsEmptySelection` (page 1556)
- `mode` (page 1572)
- `selectCellAtRow:column:` (page 1578)

Declared In
`NSMatrix.h`

doubleAction

Returns the matrix's double-click action method.

```
- (SEL)doubleAction
```

Return Value
The action method sent by the receiver to its target when the user double-clicks an entry or `NULL` if there's no double-click action.

Discussion
The double-click action of an `NSMatrix` is sent after the appropriate single-click action (for the `NSCell` clicked or for the `NSMatrix` if the `NSCell` doesn't have its own action). If there is no double-click action and the `NSMatrix` doesn't ignore multiple clicks, the single-click action is sent twice.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `action` (page 814) (NSControl)
- `target` (page 844) (NSControl)
- `ignoresMultiClick` (page 820) (NSControl)
- `sendDoubleAction` (page 1583)
- `setDoubleAction:` (page 1587)

Declared In
`NSMatrix.h`

drawCellAtRow:column:

Displays the cell at the specified row and column.

```
- (void)drawCellAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row
The row containing the cell to draw.

column
The column containing the cell to draw.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawCell:` (page 818) (NSControl)
- `drawCellInside:` (page 818) (NSControl)

Declared In
`NSMatrix.h`

drawsBackground

Returns a Boolean value indicating whether the matrix draws its background.

```
- (BOOL)drawsBackground
```

Return Value
`YES` if the receiver draws its background (the space between the cells); otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `backgroundColor` (page 1557)
- `drawsCellBackground` (page 1563)
- `setDrawsBackground:` (page 1587)

Declared In
`NSMatrix.h`

drawsCellBackground

Returns whether the matrix draws the background within each of its cells.

```
- (BOOL)drawsCellBackground
```

Return Value
`YES` if the receiver draws the cell background; otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cellBackgroundColor` (page 1558)
- `drawsBackground` (page 1563)
- `setDrawsCellBackground:` (page 1588)

Declared In
`NSMatrix.h`

getNumberOfRows:columns:

Obtains the number of rows and columns in the receiver.

```
- (void)getNumberOfRows:(NSInteger *)rowCount columns:(NSInteger *)columnCount
```

Parameters
rowCount
On return, the number of rows in the matrix.
columnCount
On return, the number of columns in the matrix.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfColumns` (page 1573)
- `numberOfRows` (page 1573)

Declared In
`NSMatrix.h`

getRow:column:forPoint:

Indicates whether the specified point lies within one of the cells of the matrix and returns the location of the cell within which the point lies.

```
- (BOOL)getRow:(NSInteger *)row column:(NSInteger *)column forPoint:(NSPoint)aPoint
```

Parameters

row
On return, the row of the cell containing the specified point.

column
On return, the column of the cell containing the specified point.

aPoint
The point to locate; this point should be in the coordinate system of the receiver.

Return Value
`YES` if the point lies within one of the cells in the receiver; `NO` if the point falls outside the bounds of the receiver or lies within an intercell spacing.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `getRow:column:ofCell:` (page 1564)

Declared In
`NSMatrix.h`

getRow:column:ofCell:

Searches the receiver for the specified cell and returns the row and column of the cell

```
- (BOOL)getRow:(NSInteger *)row column:(NSInteger *)column ofCell:(NSCell *)aCell
```

Parameters

row
On return, the row in which the cell is located.

column
On return, the column in which the cell is located.

aCell
The cell to locate within the matrix.

Return Value
`YES` if the cell is one of the cells in the receiver, `NO` otherwise.

Discussion
.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- getRow:column:forPoint:` (page 1564)

Declared In
`NSMatrix.h`

highlightCell:atRow:column:

Highlights or unhighlights the cell at the specified row and column location.

```
- (void)highlightCell:(BOOL)flag atRow:(NSInteger)row column:(NSInteger)column
```

Parameters

flag
`YES` to highlight the cell; `NO` to unhighlight the cell.

row
The row containing the cell.

column
The column containing the cell.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMatrix.h`

initWithFrame:

Initializes a newly allocated matrix with the specified frame.

```
- (id)initWithFrame:(NSRect)frameRect
```

Parameters

frameRect
The frame with which to initialize the matrix.

Return Value
The `NSMatrix`, initialized with default parameters. The new `NSMatrix` contains no rows or columns. The default mode is `NSRadioModeMatrix`. The default cell class is `NSActionCell`.

Discussion
.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSMatrix.h

initWithFrame:mode:cellClass:numberOfRows:numberOfColumns:

Initializes and returns a newly allocated matrix of the specified size using cells of the given class.

```
- (id)initWithFrame:(NSRect)frameRect mode:(NSMatrixMode)aMode
    cellClass:(Class)classId numberOfRows:(NSInteger)numRows
    numberOfColumns:(NSInteger)numColumns
```

Parameters

frameRect
The matrix's frame.

aMode
The tracking mode for the matrix; this can be one of the modes described in NSMatrixMode (page 1598).

classId
The class to use for any cells that the matrix creates and uses. This can be obtained by sending a class message to the desired subclass of NSCell.

numRows
The number of rows in the matrix.

numColumns
The number of columns in the matrix.

Return Value
The initialized instance of NSMatrix.

Discussion
This method is the designated initializer for matrices that add cells by creating instances of an NSCell subclass.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSMatrix.h

initWithFrame:mode:prototype:numberOfRows:numberOfColumns:

Initializes and returns a newly allocated matrix of the specified size using the given cell as a prototype.

```
- (id)initWithFrame:(NSRect)frameRect mode:(NSMatrixMode)aMode prototype:(NSCell
    *)aCell numberOfRows:(NSInteger)numRows numberOfColumns:(NSInteger)numColumns
```

Parameters

frameRect
The matrix's frame.

aMode
: The tracking mode for the matrix; this can be one of the modes described in `NSMatrixMode` (page 1598).

aCell
: An instance of a subclass of `NSCell`, which the new matrix copies when it creates new cells.

numRows
: The number of rows in the matrix.

numColumns
: The number of columns in the matrix.

Discussion
This method is the designated initializer for matrices that add cells by copying an instance of an `NSCell` subclass.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ButtonMadness

Declared In
`NSMatrix.h`

insertColumn:

Inserts a new column of cells at the specified location. .

```
- (void)insertColumn:(NSInteger)column
```

Parameters

column
: The number of the column before which the new column is inserted. If *column* is greater than the number of columns in the receiver, enough columns are created to expand the receiver to be *column* columns wide.

Discussion
New cells are created if needed with `makeCellAtRow:column:` (page 1571). This method redraws the receiver. Your code may need to send `sizeToCells` (page 1593) after sending this method to resize the receiver to fit the newly added cells.

If the number of rows or columns in the receiver has been changed with `renewRows:columns:` (page 1576), new cells are created only if they're needed. This fact allows you to grow and shrink an `NSMatrix` without repeatedly creating and freeing the cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addColumn` (page 1554)
- `insertRow:` (page 1568)

Declared In
`NSMatrix.h`

insertColumn:withCells:

Inserts a new column of cells before the specified column, using the given cells.

```
- (void)insertColumn:(NSInteger)column withCells:(NSArray *)newCells
```

Parameters

column

The column at which to insert the new cells.

newCells

An array of objects to use to fill the new column, starting with the object at index 0. Each object should be an instance of `NSCell` or one of its subclasses (usually `NSActionCell`).

Discussion

If *column* is greater than the number of columns in the receiver, enough columns are created to expand the receiver to be *column* columns wide. *newCells* should either be empty or contain a sufficient number of cells to fill each new column. If *newCells* is `nil` or an array with no elements, the call is equivalent to calling `insertColumn:` (page 1567). Extra cells are ignored, unless the matrix is empty. In that case, a matrix is created with one column and enough rows for all the elements of *newCells*.

This method redraws the receiver. Your code may need to send `sizeToCells` (page 1593) after sending this method to resize the receiver to fit the newly added cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addColumnWithCells:` (page 1554)
- `insertRow:withCells:` (page 1569)

Declared In

`NSMatrix.h`

insertRow:

Inserts a new row of cells before the specified row.

```
- (void)insertRow:(NSInteger)row
```

Parameters

row

The location at which to insert the new row. If this is greater than the number of rows in the receiver, enough rows are created to expand the receiver to be *row* rows high.

Discussion

New cells are created if needed with `makeCellAtRow:column:` (page 1571). This method redraws the receiver. Your code may need to send `sizeToCells` (page 1593) after sending this method to resize the receiver to fit the newly added cells.

If the number of rows or columns in the receiver has been changed with `renewRows:columns:` (page 1576), then new cells are created only if they're needed. This fact allows you to grow and shrink an `NSMatrix` without repeatedly creating and freeing the cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addRow` (page 1555)
- `insertColumn:` (page 1567)

Declared In
`NSMatrix.h`

insertRow:withCells:

Inserts a new row of cells before the specified row, using the given cells.

```
- (void)insertRow:(NSInteger)row withCells:(NSArray *)newCells
```

Parameters

row
The location at which to insert the new row.

newCells
An array of objects to use when filling the new row, starting with the object at index 0. Each object in *newCells* should be an instance of `NSCell` or one of its subclasses (usually `NSActionCell`).

Discussion
If *row* is greater than the number of rows in the receiver, enough rows are created to expand the receiver to be *row* rows high. *newCells* should either be empty or contain a sufficient number of cells to fill each new row. If *newCells* is `nil` or an array with no elements, the call is equivalent to calling `insertRow:` (page 1568). Extra cells are ignored, unless the matrix is empty. In that case, a matrix is created with one row and enough columns for all the elements of *newCells*.

This method redraws the receiver. Your code may need to send `sizeToCells` (page 1593) after sending this method to resize the receiver to fit the newly added cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addRowWithCells:` (page 1556)
- `insertColumn:withCells:` (page 1568)

Declared In
`NSMatrix.h`

intercellSpacing

Returns the spacing between cells in the matrix.

```
- (NSSize)intercellSpacing
```

Return Value
The vertical and horizontal spacing between cells in the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cellSize` (page 1559)
- `setIntercellSpacing:` (page 1588)

Declared In
`NSMatrix.h`

isAutoscroll

Returns a Boolean value indicating whether the receiver is automatically scrolled.

```
- (BOOL)isAutoscroll
```

Return Value
`YES` if the receiver will be automatically scrolled whenever the cursor is dragged outside the receiver after a mouse-down event within its bounds; otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `scrollCellToVisibleAtRow:column:` (page 1577)
- `setScrollable:` (page 1590)

Declared In
`NSMatrix.h`

isSelectionByRect

Returns a Boolean value indicating whether the user can drag the cursor to select a rectangle of cells in the matrix.

```
- (BOOL)isSelectionByRect
```

Return Value
`YES` if the user can select a rectangle of cells in the receiver by dragging the cursor, `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setSelectionFrom:to:anchor:highlight:` (page 1591)

Declared In
`NSMatrix.h`

keyCell

Returns the cell that will be clicked when the user presses the Space bar.

```
- (id)keyCell
```

Return Value
The cell that will be clicked when the user presses the Space bar.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `tabKeyTraversesCells` (page 1595)
- `setKeyCell:` (page 1589)

Declared In
`NSMatrix.h`

makeCellAtRow:column:

Creates a new cell at the location specified by the given row and column in the receiver.

```
- (NSCell *)makeCellAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row
The row in which to create the new cell.

column
The column in which to create the new cell.

Return Value
The newly created cell.

Discussion
If the receiver has a prototype cell, it's copied to create the new cell. If not, and if the receiver has a cell class set, it allocates and initializes (with `init`) an instance of that class. If the receiver hasn't had either a prototype cell or a cell class set, `makeCellAtRow:column:` creates an `NSActionCell`.

Your code should never invoke this method directly; it's used by `addRow` (page 1555) and other methods when a cell must be created. It may be overridden to provide more specific initialization of cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addColumn` (page 1554)
- `addRow` (page 1555)
- `insertColumn:` (page 1567)
- `insertRow:` (page 1568)
- `setCellClass:` (page 1585)
- `setPrototype:` (page 1589)

Declared In
NSMatrix.h

mode

Returns the selection mode of the matrix.

```
- (NSMatrixMode)mode
```

Return Value
The selection mode of the receiver. Possible return values are defined in `NSMatrixMode` (page 1598).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initWithFrame:mode:cellClass:numberOfRows:numberOfColumns:` (page 1566)
- `initWithFrame:mode:prototype:numberOfRows:numberOfColumns:` (page 1566)
- `setMode:` (page 1589)

Declared In
NSMatrix.h

mouseDown:

Responds to a mouse-down event.

```
- (void)mouseDown:(NSEvent *)theEvent
```

Parameters
theEvent
The mouse-down event.

Discussion
A mouse-down event in a text cell initiates editing mode. A double click in any cell type except a text cell sends the double-click action of the receiver (if there is one) in addition to the single-click action.

Your code should never invoke this method, but you may override it to implement different mouse tracking than `NSMatrix` does. The response of the receiver depends on its selection mode, as explained in the class description.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sendAction` (page 1582)
- `sendDoubleAction` (page 1583)

Declared In
NSMatrix.h

mouseDownFlags

Returns the flags in effect at the mouse-down event that started the current tracking session.

```
- (NSInteger)mouseDownFlags
```

Return Value
The flags in effect when the mouse-down event is generated.

Discussion
The `NSMatrixmouseDown:` (page 1572) method obtains these flags by sending a `modifierFlags` (page 1082) message to the event passed into `mouseDown:` (page 1572). Use this method if you want to access these flags. This method is valid only during tracking; it isn't useful if the target of the receiver initiates another tracking loop as part of its action method (as a cell that pops up a pop-up list does, for example).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- sendActionOn:` (page 827) (NSCell)

Declared In
`NSMatrix.h`

numberOfColumns

Returns the number of columns in the receiver.

```
- (NSInteger)numberOfColumns
```

Return Value
The number of columns in the matrix.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- getNumberOfRows:columns:` (page 1563)

Declared In
`NSMatrix.h`

numberOfRows

Returns the number of rows in the receiver.

```
- (NSInteger)numberOfRows
```

Return Value
The number of rows in the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `getNumberOfRows:columns:` (page 1563)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSMatrix.h`

performKeyEquivalent:

Looks for a cell that has the given key equivalent and, if found, makes that cell respond as if clicked.

```
- (BOOL)performKeyEquivalent:(NSEvent *)theEvent
```

Parameters

theEvent
The event containing the character for which to find a key equivalent.

Return Value
`YES` if a cell in the receiver responds to the key equivalent in *theEvent*, `NO` if no cell responds.

Discussion
If there's a cell in the receiver that has a key equivalent equal to the character in [*theEvent*`charactersIgnoringModifiers` (page 1074)] (taking into account any key modifier flags) and that cell is enabled, that cell is made to react as if the user had clicked it: by highlighting, changing its state as appropriate, sending its action if it has one, and then unhighlighting.

Your code should never send this message—it is sent when the receiver or one of its superviews is the first responder and the user presses a key. You may want to override this method to change the way key equivalents are performed or displayed or to disable them in your subclass.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMatrix.h`

prototype

Returns the prototype cell that's copied when a new cell is created.,

```
- (id)prototype
```

Return Value
The cell that the matrix copies whenever it creates a new cell, or `nil` if there is none.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initWithFrame:mode:prototype:numberOfRows:numberOfColumns:` (page 1566)
- `makeCellAtRow:column:` (page 1571)

- setPrototype: (page 1589)

Declared In
NSMatrix.h

putCell:atRow:column:

Replaces the cell at the specified row and column with the new cell.

```
- (void)putCell:(NSCell *)newCell atRow:(NSInteger)row column:(NSInteger)column
```

Parameters

newCell
The cell to insert into the matrix.

row
The row in which to place the new cell.

column
The column in which to place the new cell.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSMatrix.h

removeColumn:

Removes the specified column at from the receiver.

```
- (void)removeColumn:(NSInteger)column
```

Parameters

column
The column to remove.

Discussion
The column's cells are autoreleased. This method redraws the receiver. Your code should normally send sizeToCells (page 1593) after invoking this method to resize the receiver so it fits the reduced cell count.

Availability
Available in Mac OS X v10.0 and later.

See Also
- removeRow: (page 1576)
- addColumn (page 1554)
- insertColumn: (page 1567)

Declared In
NSMatrix.h

removeRow:

Removes the specified row from the receiver.

```
- (void)removeRow:(NSInteger)row
```

Parameters

row

The row to remove.

Discussion

The row's cells are autoreleased. This method redraws the receiver. Your code should normally send `sizeToCells` (page 1593) after invoking this method to resize the receiver so it fits the reduced cell count.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `removeColumn:` (page 1575)
- `addRow` (page 1555)
- `insertRow:` (page 1568)

Declared In

`NSMatrix.h`

renewRows:columns:

Changes the number of rows and columns in the receiver.

```
- (void)renewRows:(NSInteger)newRows columns:(NSInteger)newCols
```

Parameters

newRows

The new number of rows in the matrix.

newCols

The new number of columns in the matrix.

Discussion

This method uses the same cells as before, creating new cells only if the new size is larger; it never frees cells. It doesn't redisplay the receiver. Your code should normally send `sizeToCells` (page 1593) after invoking this method to resize the receiver so it fits the changed cell arrangement. This method deselects all cells in the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addColumn` (page 1554)
- `addRow` (page 1555)
- `removeColumn:` (page 1575)
- `removeRow:` (page 1576)

Related Sample Code

NewsReader

Declared In
NSMatrix.h

resetCursorRects

Resets cursor rectangles so the cursor becomes an I-beam over text cells.

```
- (void)resetCursorRects
```

Discussion
This method resets the cursor rectangles by sending `resetCursorRect:inView:` (page 574) to each cell in the receiver. Any cell that has a cursor rectangle to set up should then send `addCursorRect:cursor:` (page 3139) back to the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `resetCursorRect:inView:` (page 574) (NSCell)
- `addCursorRect:cursor:` (page 3139) (NSView)

Declared In
NSMatrix.h

scrollCellToVisibleAtRow:column:

Scrolls the receiver so the specified cell is visible.

```
- (void)scrollCellToVisibleAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row
The row of the cell to make visible.

column
The column of the cell to make visible.

Discussion
This method scrolls if the receiver is in a scrolling view and *row* and *column* represent a valid cell within the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `scrollRectToVisible:` (page 3210) (NSView)

Declared In
NSMatrix.h

selectAll:

Selects and highlights all cells in the receiver.

```
- (void)selectAll:(id)sender
```

Parameters

sender

This argument is ignored.

Discussion

Editable text cells and disabled cells are not selected. The receiver is redisplayed.

If the selection mode is not `NSListModeMatrix` (page 1599), this method does nothing.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `selectCell:` (page 825) (NSControl)

Declared In

`NSMatrix.h`

selectCellAtRow:column:

Selects the cell at the specified row and column within the receiver.

```
- (void)selectCellAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row

The row of the cell to select.

column

The column of the cell to select.

Discussion

If the specified cell is an editable text cell, its text is selected. If either *row* or *column* is –1, then the current selection is cleared (unless the receiver is an `NSRadioModeMatrix` and doesn't allow empty selection). This method redraws the affected cells.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `allowsEmptySelection` (page 1556)
- `mode` (page 1572)
- `selectCell:` (page 825) (NSControl)

Related Sample Code

PDFKitLinker2

Sketch+Accessibility

Sketch-112

Declared In
NSMatrix.h

selectCellWithTag:

Selects the last cell with the given tag.

```
- (BOOL)selectCellWithTag:(NSInteger)anInt
```

Parameters

anInt

The tag of the cell to select.

Return Value
YES if the receiver contains a cell whose tag matches *anInt*, or NO if no such cell exists

Discussion
If the matrix has at least one cell whose tag is equal to *anInt*, the last cell (when viewing the matrix as a row-ordered array) is selected. If the specified cell is an editable text cell, its text is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- cellWithTag: (page 1560)
- selectCell: (page 825) (NSControl)

Related Sample Code
Quartz 2D Shadings
TipWrapper

Declared In
NSMatrix.h

selectedCell

Returns the most recently selected cell.

```
- (id)selectedCell
```

Return Value
The most recently selected cell or nil if no cell is selected. If more than one cell is selected, this method returns the cell that is lowest and farthest to the right in the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSMatrix.h

selectedCells

Returns the receiver's selected and highlighted cells.

```
- (NSArray *)selectedCells
```

Return Value
An array containing all of the receiver's highlighted cells plus its selected cell.

Discussion
See the class description for a discussion of the selected cell.

As an alternative to using `setSelectionFrom:to:anchor:highlight:` (page 1591) for programmatically making discontiguous selections of cells in a matrix, you could first set the single selected cell and then set subsequent cells to be highlighted; afterwards you can call `selectedCells` (page 1580) to obtain the selection of cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHighlighted:` (page 588) (NSCell)
- `selectedCell` (page 1579)

Declared In
`NSMatrix.h`

selectedColumn

Returns the column of the selected cell.

```
- (NSInteger)selectedColumn
```

Return Value
The column number of the selected cell or –1 if no cells are selected. If cells in multiple columns are selected, this method returns the number of the last (rightmost) column containing a selected cell.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer Live DV

Declared In
`NSMatrix.h`

selectedRow

Returns the row of the selected cell.

```
- (NSInteger)selectedRow
```

Return Value
the row number of the selected cell, or –1 if no cells are selected. If cells in multiple rows are selected, this method returns the number of the last row containing a selected cell.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Sketch+Accessibility
Sketch-112

Declared In
NSMatrix.h

selectText:

Selects text in the currently selected cell or in the key cell.

```
- (void)selectText:(id)sender
```

Discussion
If the currently selected cell is editable and enabled, its text is selected. Otherwise, the key cell is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- keyCell (page 1571)
- selectText: (page 2801) (NSTextField)

Declared In
NSMatrix.h

selectTextAtRow:column:

Selects the text in the cell at the specified location and returns the cell.

```
- (id)selectTextAtRow:(NSInteger)row column:(NSInteger)column
```

Parameters

row
The row containing the text to select.

column
The column containing the text to select.

Return Value
If it is both editable and selectable, the cell at the specified row and column. If the cell at the specified location, is either not editable or not selectable, this method does nothing and returns nil. If *row* and *column* indicate a cell that is outside the receiver, this method does nothing and returns the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectText:` (page 1581)

Declared In
`NSMatrix.h`

sendAction

If the selected cell has both an action and a target, sends its action to its target.

```
- (BOOL)sendAction
```

Return Value
`YES` if an action was successfully sent to a target. If the selected cell is disabled, this method does nothing and returns `NO`.

Discussion
If the cell has an action but no target, its action is sent to the target of the receiver. If the cell doesn't have an action, or if there is no selected cell, the receiver sends its own action to its target.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sendDoubleAction` (page 1583)
- `action` (page 544) (NSCell)
- `target` (page 608) (NSCell)

Declared In
`NSMatrix.h`

sendAction:to:forAllCells:

Iterates through the cells in the receiver, sending the specified selector to an object for each cell.

```
- (void)sendAction:(SEL)aSelector to:(id)anObject forAllCells:(BOOL)flag
```

Parameters

aSelector
The selector to send to the object for each cell. This must represent a method that takes a single argument: the `id` of the current cell in the iteration. *aSelector*'s return value must be a `BOOL`. If *aSelector* returns `NO` for any cell, `sendAction:to:forAllCells:` terminates immediately, without sending the message for the remaining cells. If it returns `YES`, `sendAction:to:forAllCells:` proceeds to the next cell.

anObject
The object that is sent the selector for each cell in the matrix.

flag
`YES` if the method should iterate through all cells in the matrix; `NO` if it should iterate through just the selected cells in the matrix.

Discussion
Iteration begins with the cell in the upper-left corner of the receiver, proceeding through the appropriate entries in the first row, then on to the next.

This method is not invoked to send action messages to target objects in response to mouse-down events in the receiver. Instead, you can invoke it if you want to have multiple cells in an `NSMatrix` interact with an object. For example, you could use it to verify the titles in a list of items or to enable a series of radio buttons based on their purpose in relation to *`anObject`*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMatrix.h`

sendDoubleAction

Sends the double-click action message to the target of the receiver.

```
- (void)sendDoubleAction
```

Discussion
If the receiver doesn't have a double-click action, the double-click action message of the selected cell (as returned by `selectedCell` (page 1579)) is sent to the selected cell's target. Finally, if the selected cell also has no action, then the single-click action of the receiver is sent to the target of the receiver.

If the selected cell is disabled, this method does nothing.

Your code shouldn't invoke this method; it's sent in response to a double-click event in the `NSMatrix`. Override it if you need to change the search order for an action to send.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sendAction` (page 1582)
- `ignoresMultiClick` (page 820) (NSControl)

Declared In
`NSMatrix.h`

setAllowsEmptySelection:

Sets whether a radio-mode matrix allows an empty selection.

```
- (void)setAllowsEmptySelection:(BOOL)flag
```

Parameters
`flag`

`YES` to make the receiver allow one or zero cells to be selected. `NO` if the receiver should allow one and only one cell (not zero cells) to be selected. This setting has effect only in the `NSRadioModeMatrix` selection mode.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsEmptySelection` (page 1556)

Declared In
`NSMatrix.h`

setAutoscroll:

Sets whether the receiver is automatically scrolled.

```
- (void)setAutoscroll:(BOOL)flag
```

Parameters

flag

`YES` to indicate that the receiver, if it is in a scrolling view, should be automatically scrolled whenever the cursor is dragged outside the receiver after a mouse-down event within the bounds of the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMatrix.h`

setAutosizesCells:

Sets whether the cell sizes change when the receiver is resized.

```
- (void)setAutosizesCells:(BOOL)flag
```

Parameters

flag

`YES` to specify that, whenever the receiver is resized, the sizes of the cells change in proportion, keeping the intercell space constant; further, this method verifies that the cell sizes and intercell spacing add up to the exact size of the receiver, adjusting the size of the cells and updating the receiver if they don't. If *flag* is `NO`, then the intercell spacing and cell size remain constant.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `autosizesCells` (page 1556)

Declared In
`NSMatrix.h`

setBackgroundColor:

Sets the background color for the receiver and redraws the receiver.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Parameters

aColor

The background color used to fill the space between cells or the space behind any non-opaque cells. The default background color is the color returned by the `NSColor` method `controlColor` (page 678).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawsBackground` (page 1563)
- `setCellBackgroundColor:` (page 1585)
- `backgroundColor` (page 1557)

Declared In
`NSMatrix.h`

setCellBackgroundColor:

Sets the background color for the cells in the receiver

```
- (void)setCellBackgroundColor:(NSColor *)aColor
```

Parameters

aColor

The background color used to fill the space behind non-opaque cells. The default cell background color is the color returned by the `NSColor` method `controlColor` (page 678)

Discussion
.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawsCellBackground` (page 1563)
- `setBackgroundColor:` (page 1584)
- `cellBackgroundColor` (page 1558)

Declared In
`NSMatrix.h`

setCellClass:

Configures the receiver to use instances of the specified class when creating new cells.

```
- (void)setCellClass:(Class)aClass
```

Parameters

aClass

The class to use when creating new cells. This should be the `id` of a subclass of `NSCell`, which can be obtained by sending the `class` message to either the `NSCell` subclass object or to an instance of that subclass. The default cell class is that set with the class method `setCellClass:` (page 813), or `NSActionCell` if no other default cell class has been specified.

Discussion

You need to use this method only with matrices initialized with `initWithFrame:` (page 1565), because the other initializers allow you to specify an instance-specific cell class or cell prototype.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addColumn` (page 1554)
- `addRow` (page 1555)
- `insertColumn:` (page 1567)
- `insertRow:` (page 1568)
- `makeCellAtRow:column:` (page 1571)
- `setPrototype:` (page 1589)
- `cellClass` (page 1558)

Declared In

`NSMatrix.h`

setCellSize:

Sets the width and height of each of the cells in the matrix.

```
- (void)setCellSize:(NSSize)aSize
```

Parameters

aSize

The new width and height of cells in the receiver.

Discussion

This method may change the size of the receiver. It does not redraw the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `calcSize` (page 816) (NSControl)
- `cellSize` (page 1559)

Declared In

`NSMatrix.h`

setDelegate:

Sets the delegate for messages from the field editor.

```
- (void)setDelegate:(id < NSMatrixDelegate >)anObject
```

Parameters

anObject

The delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textShouldBeginEditing:` (page 1597)
- `textShouldEndEditing:` (page 1597)
- `delegate` (page 1560)

Declared In
`NSMatrix.h`

setDoubleAction:

Sets the action sent to the target of the receiver when the user double-clicks a cell.

```
- (void)setDoubleAction:(SEL)aSelector
```

Parameters

aSelector

The selector to make the double-click action of the receiver.

Discussion
A double-click action is always sent after the appropriate single-click action, which is the cell's single-click action, if it has one, or the receiver single-click action, otherwise. If *aSelector* is a non-`NULL` selector, this method also sets the *ignoresMultiClick* flag to `NO`; otherwise, it leaves the flag unchanged.

If an `NSMatrix` has no double-click action set, then by default a double click is treated as a single click.

For the method to have any effect, the receiver's action and target must be set to the class in which the selector is declared. See *Action Messages* for additional information on action messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sendDoubleAction` (page 1583)
- `setAction:` (page 828) (NSControl)
- `setTarget:` (page 838) (NSControl)
- `doubleAction` (page 1562)

Declared In
`NSMatrix.h`

setDrawsBackground:

Sets whether the receiver draws its background.

```
- (void)setDrawsBackground:(BOOL)flag
```

Parameters

flag

YES if the receiver should draw its background (the space between the cells); NO if it should not.

Availability
Available in Mac OS X v10.0 and later.

See Also
- backgroundColor (page 1557)
- setDrawsCellBackground: (page 1588)
- drawsBackground (page 1563)

Declared In
NSMatrix.h

setDrawsCellBackground:

Sets whether the receiver draws the background within each of its cells.

```
- (void)setDrawsCellBackground:(BOOL)flag
```

Parameters

flag

YES if the receiver should draw the background in its cells; NO if it should not.

Availability
Available in Mac OS X v10.0 and later.

See Also
- cellBackgroundColor (page 1558)
- setDrawsBackground: (page 1587)
- drawsCellBackground (page 1563)

Declared In
NSMatrix.h

setIntercellSpacing:

Sets the spacing between cells in the matrix.

```
- (void)setIntercellSpacing:(NSSize)aSize
```

Parameters

aSize

The vertical and horizontal spacing to use between cells in the receiver. By default, both values are 1.0 in the receiver's coordinate system.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cellSize` (page 1559)
- `intercellSpacing` (page 1569)

Declared In
`NSMatrix.h`

setKeyCell:

Sets the cell that will be clicked when the user presses the Space bar.

```
- (void)setKeyCell:(NSCell *)aCell
```

Parameters

aCell

The cell to set as the key cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTabKeyTraversesCells:` (page 1592)
- `keyCell` (page 1571)

Declared In
`NSMatrix.h`

setMode:

Sets the selection mode of the receiver.

```
- (void)setMode:(NSMatrixMode)aMode
```

Parameters

aMode

The selection mode of the matrix. Possible values are listed in `NSMatrixMode` (page 1598).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initWithFrame:mode:cellClass:numberOfRows:numberOfColumns:` (page 1566)
- `initWithFrame:mode:prototype:numberOfRows:numberOfColumns:` (page 1566)
- `mode` (page 1572)

Declared In
`NSMatrix.h`

setPrototype:

Sets the prototype cell that's copied whenever the matrix creates a new cell.

```
- (void)setPrototype:(NSCell *)aCell
```

Parameters

aCell

The cell to copy when creating new cells.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initWithFrame:mode:prototype:numberOfRows:numberOfColumns:` (page 1566)
- `makeCellAtRow:column:` (page 1571)
- `prototype` (page 1574)

Declared In
NSMatrix.h

setScrollable:

Specifies whether the cells in the matrix are scrollable.

```
- (void)setScrollable:(BOOL)flag
```

Parameters

flag

`YES` to make all the cells in the receiver scrollable, so the text they contain scrolls to remain in view if the user types past the edge of the cell. If *flag* is `NO`, all cells are made nonscrolling. The prototype cell, if there is one, is also set accordingly

Availability
Available in Mac OS X v10.0 and later.

See Also
- `prototype` (page 1574)
- `setScrollable:` (page 594) (NSCell)

Declared In
NSMatrix.h

setSelectionByRect:

Sets whether the user can select a rectangle of cells in the receiver by dragging the cursor.

```
- (void)setSelectionByRect:(BOOL)flag
```

Parameters

flag

`YES` if the matrix should allow the user to select a rectangle of cells by dragging. `NO` if selection in the matrix should be on a row-by-row basis. The default is `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setSelectionFrom:to:anchor:highlight:` (page 1591)

Declared In
`NSMatrix.h`

setSelectionFrom:to:anchor:highlight:

Programmatically selects a range of cells.

```
- (void)setSelectionFrom:(NSInteger)startPos to:(NSInteger)endPos
    anchor:(NSInteger)anchorPos highlight:(BOOL)lit
```

Parameters

startPos
The position of the cell that marks where the user would have pressed the mouse button.

endPos
The position of the cell that marks where the user would have released the mouse button.

anchorPos
The position of the cell to treat as the last cell the user would have selected. To simulate Shift-dragging (continuous selection) *anchorPos* should be the *endPos* used in the last method call. To simulate Command-dragging (discontinuous selection), *anchorPos* should be the same as this method call's *startPos*.

lit
`YES` if cells selected by this method should be highlighted.

Discussion
startPos, *endPos*, and *anchorPos* are cell positions, counting from 0 at the upper left cell of the receiver, in row order. For example, the third cell in the top row would be number 2.

To simulate dragging without a modifier key, deselecting anything that was selected before, call `deselectAllCells` (page 1561) before calling this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isSelectionByRect` (page 1570)
- `selectedCells` (page 1580)

Declared In
`NSMatrix.h`

setState:atRow:column:

Sets the state of the cell at specified location.

```
- (void)setState:(NSInteger)value atRow:(NSInteger)row column:(NSInteger)column
```

Parameters

value
The value to assign to the cell.

row
The row in which the cell is located.

column
The column in which the cell is located.

Discussion
For radio-mode matrices, if *value* is nonzero the specified cell is selected before its state is set to *value*. If *value* is 0 and the receiver is a radio-mode matrix, the currently selected cell is deselected (providing that empty selection is allowed).

This method redraws the affected cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsEmptySelection` (page 1556)
- `setState:` (page 596) (NSCell)
- `selectCellAtRow:column:` (page 1578)

Declared In
`NSMatrix.h`

setTabKeyTraversesCells:

Sets whether pressing the Tab key advances the key cell to the next selectable cell.

```
- (void)setTabKeyTraversesCells:(BOOL)flag
```

Parameters

flag
`YES` if pressing the Tab key should advance the key cell to the next selectable cell in the receiver. If this is `NO` or if there aren't any selectable cells after the current one, the next view in the window becomes key when the user presses the Tab key.

Discussion
Pressing Shift-Tab causes the key cell to advance in the opposite direction (if *flag* is `NO`, or if there aren't any selectable cells before the current one, the previous view in the window becomes key).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectKeyViewFollowingView:` (page 3364) (NSWindow)
- `selectNextKeyView:` (page 3364) (NSWindow)
- `setKeyCell:` (page 1589)
- `tabKeyTraversesCells` (page 1595)

Declared In
`NSMatrix.h`

setToolTip:forCell:

Sets the tooltip for the cell.

```
- (void)setToolTip:(NSString *)toolTipString forCell:(NSCell *)cell
```

Parameters

toolTipString

The string to use as the cell's tooltip (or help tag).

cell

The cell to which to assign the tooltip.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `toolTipForCell:` (page 1598)

Declared In
`NSMatrix.h`

setValidateSize:

Specifies whether the receiver's size information is validated.

```
- (void)setValidateSize:(BOOL)flag
```

Parameters

flag

`YES` to assume that the size information in the receiver is correct. If *flag* is `NO`, the `NSControl` method `calcSize` (page 816) will be invoked before any further drawing is done.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMatrix.h`

sizeToCells

Changes the width and the height of the receiver's frame so it exactly contains the cells.

```
- (void)sizeToCells
```

Discussion
This method does not redraw the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFrameSize:` (page 3221) (NSView)
- `sizeToFit` (page 839) (NSControl)

Declared In
NSMatrix.h

sortUsingFunction:context:

Sorts the receiver's cells in ascending order as defined by the specified comparison function.

```
- (void)sortUsingFunction:(NSInteger (*)(id, id, void *))comparator context:(void
    *)context
```

Parameters

comparator

The function to use when comparing cells. The comparison function is used to compare two elements at a time and should return `NSOrderedAscending` if the first element is smaller than the second, `NSOrderedDescending` if the first element is larger than the second, and `NSOrderedSame` if the elements are equal.

context

Context passed to the comparison function as its third argument, each time its called. This allows the comparison to be based on some outside parameter, such as whether character sorting is case-sensitive or case-insensitive.

Availability
Available in Mac OS X v10.0 and later.

See Also
`sortUsingFunction:context:` (NSMutableArray)

Declared In
NSMatrix.h

sortUsingSelector:

Sorts the receiver's cells in ascending order as defined by the comparison method.

```
- (void)sortUsingSelector:(SEL)comparator
```

Parameters

comparator

The comparison method.

Discussion
The comparator message is sent to each object in the matrix and has as its single argument another object in the array. The comparison method is used to compare two elements at a time and should return `NSOrderedAscending` if the receiver is smaller than the argument, `NSOrderedDescending` if the receiver is larger than the argument, and `NSOrderedSame` if they are equal.

Availability
Available in Mac OS X v10.0 and later.

See Also
`sortUsingSelector:` (NSMutableArray)

Declared In
NSMatrix.h

tabKeyTraversesCells

Returns a Boolean value indicating whether pressing the Tab key advances the key cell to the next selectable cell.

```
- (BOOL)tabKeyTraversesCells
```

Return Value
YES if pressing the Tab key advances the key cell to the next selectable cell in the receiver; otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- keyCell (page 1571)
- setTabKeyTraversesCells: (page 1592)

Declared In
NSMatrix.h

textDidBeginEditing:

Invoked when there's a change in the text after the receiver gains first responder status.

```
- (void)textDidBeginEditing:(NSNotification *)notification
```

Parameters
notification
The NSControlTextDidBeginEditingNotification (page 847) notification.

Discussion
This method's default behavior is to post an NSControlTextDidBeginEditingNotification (page 847) along with the receiving object to the default notification center. The posted notification's user info contains the contents of notification's user info dictionary, plus an additional key-value pair. The additional key is "NSFieldEditor"; the value for this key is the text object that began editing.

Availability
Available in Mac OS X v10.0 and later.

See Also
- textDidChange: (page 1596)
- textDidEndEditing: (page 1596)
- textShouldEndEditing: (page 1597)

Declared In
NSMatrix.h

textDidChange:

Invoked when a key-down event or paste operation occurs that changes the receiver's contents.

```
- (void)textDidChange:(NSNotification *)notification
```

Parameters

notification

The NSControlTextDidChangeNotification (page 848) notification.

Discussion

This method's default behavior is to pass this message on to the selected cell (if the selected cell responds to textDidChange:) and then to post an NSControlTextDidChangeNotification (page 848) along with the receiving object to the default notification center. The posted notification's user info contains the contents of notification's user info dictionary, plus an additional key-value pair. The additional key is "NSFieldEditor"; the value for this key is the text object that changed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- textDidBeginEditing: (page 1595)
- textDidEndEditing: (page 1596)

Declared In

NSMatrix.h

textDidEndEditing:

Invoked when text editing ends.

```
- (void)textDidEndEditing:(NSNotification *)notification
```

Parameters

notification

The NSControlTextDidEndEditingNotification (page 848) notification.

Discussion

This method's default behavior is to post an NSControlTextDidEndEditingNotification (page 848) along with the receiving object to the default notification center. The posted notification's user info contains the contents of notification's user info dictionary, plus an additional key-value pair. The additional key is "NSFieldEditor"; the value for this key is the text object that began editing. After posting the notification, textDidEndEditing: sends an endEditing: (page 555) message to the selected cell, draws and makes the selected cell key, and then takes the appropriate action based on which key was used to end editing (Return, Tab, or Back-Tab).

Availability

Available in Mac OS X v10.0 and later.

See Also

- textDidBeginEditing: (page 1595)
- textDidChange: (page 1596)
- textShouldEndEditing: (page 1597)

Declared In
NSMatrix.h

textShouldBeginEditing:

Requests permission to begin editing text.

```
- (BOOL)textShouldBeginEditing:(NSText *)textObject
```

Parameters

textObject
The text object requesting permission to begin editing.

Return Value
YES if the text object should proceed to make changes. If the delegate returns NO, the text object abandons the editing operation.

The default behavior of this method is to return the value obtained from control:textShouldBeginEditing:, unless the delegate doesn't respond to that method, in which case it returns YES, thereby allowing text editing to proceed.

Discussion
This method is invoked to let the NSTextField respond to impending changes to its text. This method's default behavior is to send control:textShouldBeginEditing: to the receiver's delegate (passing the receiver and *textObject* as parameters).

Availability
Available in Mac OS X v10.0 and later.

See Also
- delegate (page 1560)

Declared In
NSMatrix.h

textShouldEndEditing:

Requests permission to end editing.

```
- (BOOL)textShouldEndEditing:(NSText *)textObject
```

Parameters

textObject
The text object requesting permission to end editing.

Return Value
YES if the text object should proceed to finish editing and resign first responder status. If the delegate returns NO, the text object selects all of its text and remains the first responder.

The textShouldEndEditing: method returns NO if the text field contains invalid contents; otherwise it returns the value passed back from control:textShouldEndEditing:.

Discussion
This method is invoked to let the `NSTextField` respond to impending loss of first-responder status. This method's default behavior checks the text field for validity; providing that the field contents are deemed valid, and providing that the delegate responds, `control:textShouldEndEditing:` is sent to the receiver's delegate (passing the receiver and *`textObject`* as parameters).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- delegate` (page 1560)

Declared In
`NSMatrix.h`

toolTipForCell:

Returns the tooltip for the specified cell.

```
- (NSString *)toolTipForCell:(NSCell *)cell
```

Parameters
`cell`
The cell for which to return the tooltip.

Return Value
The string used as the cell's tooltip.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setToolTip:forCell:` (page 1593)

Declared In
`NSMatrix.h`

Constants

NSMatrixMode

These constants determine how `NSCell` objects behave when an `NSMatrix` object is tracking the mouse.

```
typedef enum _NSMatrixMode {
    NSRadioModeMatrix     = 0,
    NSHighlightModeMatrix = 1,
    NSListModeMatrix      = 2,
    NSTrackModeMatrix     = 3
} NSMatrixMode;
```

Constants

`NSTrackModeMatrix`

The `NSCell` objects are asked to track the mouse with `trackMouse:inRect:ofView:untilMouseUp:` (page 610) whenever the cursor is inside their bounds. No highlighting is performed.

Available in Mac OS X v10.0 and later.

Declared in `NSMatrix.h`.

`NSHighlightModeMatrix`

An `NSCell` is highlighted before it's asked to track the mouse, then unhighlighted when it's done tracking.

Available in Mac OS X v10.0 and later.

Declared in `NSMatrix.h`.

`NSRadioModeMatrix`

Selects no more than one `NSCell` at a time.

Any time an `NSCell` is selected, the previously selected `NSCell` is unselected.

Available in Mac OS X v10.0 and later.

Declared in `NSMatrix.h`.

`NSListModeMatrix`

`NSCell` objects are highlighted, but don't track the mouse.

Available in Mac OS X v10.0 and later.

Declared in `NSMatrix.h`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSMatrix.h`

NSMenu Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSMenu.h
Companion guide	Application Menu and Pop-up List Programming Topics
Related sample code	GLUT MenuItemView MenuMadness PDFKitLinker2 QTAudioContextInsert

Overview

The `NSMenu` class defines an object that manages an application's menus.

Tasks

Managing the Menu Bar

`+ menuBarVisible` (page 1606)
Returns a Boolean value that indicates whether the menu bar is visible.

`+ setMenuBarVisible:` (page 1608)
Sets whether the menu bar is visible and selectable by the user.

`- menuBarHeight` (page 1622)
Returns the menu bar height for the current application's main menu.

Creating an NSMenu Object

- `initWithTitle:` (page 1617)
  Initializes and returns a menu having the specified title and with autoenabling of menu items turned on.

Adding and Removing Menu Items

- `insertItem:atIndex:` (page 1618)
  Inserts a menu item into the receiver at a specific location.
- `insertItemWithTitle:action:keyEquivalent:atIndex:` (page 1618)
  Creates and adds a menu item at a specified location in the receiver.
- `addItem:` (page 1609)
  Adds a menu item to the end of the receiver.
- `addItemWithTitle:action:keyEquivalent:` (page 1610)
  Creates a new menu item and adds it to the end of the receiver.
- `removeItem:` (page 1627)
  Removes a menu item from the receiver.
- `removeItemAtIndex:` (page 1628)
  Removes the menu item at a specified location in the receiver.
- `itemChanged:` (page 1621)
  Invoked when a menu item is modified visually (for example, its title changes).
- `removeAllItems` (page 1627)
  Removes all the menu items in the receiver.

Finding Menu Items

- `itemWithTag:` (page 1621)
  Returns the first menu item in the receiver with the specified tag.
- `itemWithTitle:` (page 1622)
  Returns the first menu item in the receiver with a specified title.
- `itemAtIndex:` (page 1620)
  Returns the menu item at a specific location of the receiver.
- `numberOfItems` (page 1624)
  Returns the number of menu items in the receiver, including separator items.
- `itemArray` (page 1620)
  Returns an array containing the receiver's menu items.

Finding Indices of Menu Items

- `indexOfItem:` (page 1614)
  Returns the index identifying the location of a specified menu item in the receiver.

- `indexOfItemWithTitle:` (page 1616)
  Returns the index of the first menu item in the receiver that has a specified title.
- `indexOfItemWithTag:` (page 1615)
  Returns the index of the first menu item in the receiver identified by a tag.
- `indexOfItemWithTarget:andAction:` (page 1616)
  Returns the index of the first menu item in the receiver that has a specified action and target.
- `indexOfItemWithRepresentedObject:` (page 1615)
  Returns the index of the first menu item in the receiver that has a given represented object.
- `indexOfItemWithSubmenu:` (page 1615)
  Returns the index of the menu item in the receiver with the given submenu.

Managing Submenus

- `setSubmenu:forItem:` (page 1632)
  Assigns a menu to be a submenu of the receiver controlled by a given menu item.
- `submenuAction:` (page 1634)
  The action method assigned to menu items that open submenus.
- `attachedMenu` (page 1611)
  Returns the menu currently attached to the receiver.
- `isAttached` (page 1619)
  Returns a Boolean value that indicates whether the receiver is currently attached to another menu.
- `locationForSubmenu:` (page 1622)
  Returns the location in screen coordinates where the given submenu is displayed when opened as a submenu of the receiver.
- `supermenu` (page 1634)
  Returns the receiver's supermenu.
- `setSupermenu:` (page 1632)
  Sets the receiver's supermenu.
- `isTornOff` (page 1619)
  Returns a Boolean value that indicates whether the receiver is offscreen or attached to another menu (or if it's the main menu).

Enabling and Disabling Menu Items

- `autoenablesItems` (page 1611)
  Returns a Boolean value that indicates whether the receiver automatically enables and disables its menu items.
- `setAutoenablesItems:` (page 1629)
  Controls whether the receiver automatically enables and disables its menu items based on delegates implementing the `NSMenuValidation` informal protocol.
- `update` (page 1635)
  Enables or disables the receiver's menu items based on the `NSMenuValidation` informal protocol and sizes the menu to fit its current menu items if necessary.

Getting and Setting the Menu Font

- `font` (page 1613)
  Returns the font used to display the menu and its submenus.
- `setFont:` (page 1630)
  Sets the font used to display the menu and its submenus.

Handling Keyboard Equivalents

- `performKeyEquivalent:` (page 1625)
  Performs the action for the menu item that corresponds to the given key equivalent.

Simulating Mouse Clicks

- `performActionForItemAtIndex:` (page 1624)
  Causes the application to send the action message of a specified menu item to its target.

Managing the Title

- `setTitle:` (page 1633)
  Sets the receiver's title.
- `title` (page 1635)
  Returns the receiver's title.

Configuring Menu Size

- `minimumWidth` (page 1624)
  Returns the minimum width of the menu.
- `setMinimumWidth:` (page 1631)
  Set the minimum width of the menu.
- `size` (page 1633)
  Returns the size of the menu.
- `sizeToFit` (page 1634)
  Resizes the receiver to exactly fit its items.

Getting Menu Properties

- `propertiesToUpdate` (page 1626)
  Returns the available properties for the menu.

Managing Menu Change Notifications

- `menuChangedMessagesEnabled` (page 1623)
  Returns a Boolean value that indicates whether messages are sent to the application's windows upon each change to the receiver.
- `setMenuChangedMessagesEnabled:` (page 1630)
  Controls whether the receiver sends messages to the application's windows upon each menu change.

Displaying Contextual Menus

- `allowsContextMenuPlugIns` (page 1611)
  Returns whether the popup menu allows appending of contextual menu plugin items.
- `setAllowsContextMenuPlugIns:` (page 1628)
  Sets whether the popup menu allows appending of contextual menu plugin items.

Displaying Context-Sensitive Help

+ `popUpContextMenu:withEvent:forView:` (page 1607)
  Displays a contextual menu over a view for an event.
+ `popUpContextMenu:withEvent:forView:withFont:` (page 1608)
  Displays a contextual menu over a view for an event using a specified font.
- `helpRequested:` (page 1613)
  Overridden by subclasses to implement specialized context-sensitive help behavior.
- `popUpMenuPositioningItem:atLocation:inView:` (page 1625)
  Pops up the menu at the specified location.

Managing Display of the State Column

- `setShowsStateColumn:` (page 1631)
  Sets whether the receiver displays the state column.
- `showsStateColumn` (page 1633)
  Returns a Boolean value that indicates whether the receiver displays the state column.

Controlling Allocation Zones

+ `menuZone` (page 1607)
  Returns the zone from which `NSMenu` objects should be allocated.
+ `setMenuZone:` (page 1609) Deprecated in Mac OS X v10.2
  Sets the zone from which `NSMenu` objects should be allocated

Handling Highlighting

- `highlightedItem` (page 1614)
  Returns the highlighted item in the receiver.

Managing the Delegate

- `setDelegate:` (page 1629)
  Sets the receiver's delegate.
- `delegate` (page 1613)
  Returns the receiver's delegate.

Handling Tracking

- `cancelTracking` (page 1612)
  Dismisses the menu and ends all menu tracking.
- `cancelTrackingWithoutAnimation` (page 1612)
  Dismisses the menu and ends all menu tracking without displaying the associated animation.

Deprecated Methods

- `contextMenuRepresentation` (page 1612) Deprecated in Mac OS X v10.2
  Deprecated. (Deprecated. Mac OS X does not use menu representations to draw menus.)
- `menuRepresentation` (page 1623) Deprecated in Mac OS X v10.2
  Deprecated. (Deprecated. Mac OS X does not use menu representations to draw menus.)
- `setContextMenuRepresentation:` (page 1629) Deprecated in Mac OS X v10.2
  Deprecated. (Deprecated. Mac OS X does not use menu representations to draw menus.)
- `setMenuRepresentation:` (page 1631) Deprecated in Mac OS X v10.2
  Deprecated. (Deprecated. Mac OS X does not use menu representations to draw menus.)
- `setTearOffMenuRepresentation:` (page 1632) Deprecated in Mac OS X v10.2
  Deprecated. (Deprecated. Mac OS X does not use menu representations to draw menus.)
- `tearOffMenuRepresentation` (page 1635) Deprecated in Mac OS X v10.2
  Deprecated. (Deprecated. Mac OS X does not use menu representations to draw menus.)

Class Methods

menuBarVisible

Returns a Boolean value that indicates whether the menu bar is visible.

```
+ (BOOL)menuBarVisible
```

Return Value
`YES` if the menu bar is visible, otherwise `NO`.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ setMenuBarVisible:` (page 1608)

Declared In
`NSMenu.h`

menuZone

Returns the zone from which `NSMenu` objects should be allocated.

```
+ (NSZone *)menuZone
```

Return Value
The zone from which `NSMenu` objects should be allocated.

Discussion
The zone is created if necessary.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DesktopImage
GLUT
MenuItemView
MenuMadness
ObjectPath

Declared In
`NSMenu.h`

popUpContextMenu:withEvent:forView:

Displays a contextual menu over a view for an event.

```
+ (void)popUpContextMenu:(NSMenu *)menu withEvent:(NSEvent *)event forView:(NSView
    *)view
```

Parameters

menu
The menu object to use for the contextual menu.

event
An `NSEvent` object representing the event.

view
The view object over which to display the contextual menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ popUpContextMenu:withEvent:forView:withFont:` (page 1608)

Related Sample Code
GLUT

Declared In
`NSMenu.h`

popUpContextMenu:withEvent:forView:withFont:

Displays a contextual menu over a view for an event using a specified font.

```
+ (void)popUpContextMenu:(NSMenu *)menu withEvent:(NSEvent *)event forView:(NSView
    *)view withFont:(NSFont *)font
```

Parameters

menu
The menu object to use for the contextual menu.

event
An `NSEvent` object representing the event.

view
The view object over which to display the contextual menu.

font
An `NSFont` object representing the font for the contextual menu. If you pass in `nil` for the font, the method uses the default font for *menu*.

Discussion
Specifying a font using the font parameter is discouraged. Instead set the menu's font using the `setFont:` (page 1630) method and pass `nil` for the *font* parameter.

Availability
Available in Mac OS X v10.3 and later.

See Also
`+ popUpContextMenu:withEvent:forView:` (page 1607)
`- popUpMenuPositioningItem:atLocation:inView:` (page 1625)

Declared In
`NSMenu.h`

setMenuBarVisible:

Sets whether the menu bar is visible and selectable by the user.

```
+ (void)setMenuBarVisible:(BOOL)visible
```

Parameters

visible

YES if menu bar is to be visible, otherwise NO.

Availability
Available in Mac OS X v10.2 and later.

See Also
+ menuBarVisible (page 1606)

Declared In
NSMenu.h

setMenuZone:

Sets the zone from which NSMenu objects should be allocated (Deprecated in Mac OS X v10.2.)

```
+ (void)setMenuZone:(NSZone *)zone
```

Parameters

zone

The memory zone to set.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSMenu.h

Instance Methods

addItem:

Adds a menu item to the end of the receiver.

```
- (void)addItem:(NSMenuItem *)newItem
```

Parameters

newItem

The menu item (an object conforming to the NSMenuItem protocol) to add to the menu.

Discussion
This method invokes insertItem:atIndex: (page 1618). Thus, the receiver does not accept the menu item if it already belongs to another menu. After adding the menu item, the receiver updates itself.

Availability
Available in Mac OS X v10.0 and later.

See Also
- addItemWithTitle:action:keyEquivalent: (page 1610)

- `removeItem:` (page 1627)
- `removeItemAtIndex:` (page 1628)

Related Sample Code
MenuItemView
MenuMadness
PDFKitLinker2
QTAudioContextInsert
QTAudioExtractionPanel

Declared In
`NSMenu.h`

addItemWithTitle:action:keyEquivalent:

Creates a new menu item and adds it to the end of the receiver.

```
- (NSMenuItem *)addItemWithTitle:(NSString *)aString action:(SEL)aSelector
    keyEquivalent:(NSString *)keyEquiv
```

Parameters

aString
A string to be made the title of the menu item.

aSelector
The action-message selector to assign to the menu item.

keyEquiv
A string identifying the key to use as a key equivalent for the menu item. If you do not want the menu item to have a key equivalent, *keyEquiv* should be an empty string (`@""`) and not `nil`.

Return Value
The created menu item (an object conforming to the `NSMenuItem` protocol) or `nil` if the object couldn't be created.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addItem:` (page 1609)
- `removeItem:` (page 1627)
- `removeItemAtIndex:` (page 1628)

Related Sample Code
ClockControl
UIElementInspector

Declared In
`NSMenu.h`

allowsContextMenuPlugIns

Returns whether the popup menu allows appending of contextual menu plugin items.

```
- (BOOL)allowsContextMenuPlugIns
```

Return Value
`YES` if the popup menu allows appending of contextual menu plugin items, otherwise `NO`.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setAllowsContextMenuPlugIns:` (page 1628)

Declared In
`NSMenu.h`

attachedMenu

Returns the menu currently attached to the receiver. (Deprecated in Mac OS X v10.2.)

```
- (NSMenu *)attachedMenu
```

Return Value
The menu currently attached to the receiver or `nil` if there's no such object.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

See Also
`- isAttached` (page 1619)

Declared In
`NSMenu.h`

autoenablesItems

Returns a Boolean value that indicates whether the receiver automatically enables and disables its menu items.

```
- (BOOL)autoenablesItems
```

Return Value
`YES` if the receiver automatically enables and disables its menu items (based on the `NSMenuValidation` informal protocol), otherwise `NO`.

Discussion
By default, `NSMenu` objects autoenable their menu items. See the protocol specification for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAutoenablesItems:` (page 1629)

Declared In
`NSMenu.h`

cancelTracking

Dismisses the menu and ends all menu tracking.

```
- (void)cancelTracking
```

Availability
Available in Mac OS X v10.5 and later.

See Also
- `cancelTrackingWithoutAnimation` (page 1612)

Related Sample Code
MenuItemView

Declared In
`NSMenu.h`

cancelTrackingWithoutAnimation

Dismisses the menu and ends all menu tracking without displaying the associated animation.

```
- (void)cancelTrackingWithoutAnimation
```

Availability
Available in Mac OS X v10.6 and later.

See Also
- `cancelTracking` (page 1612)

Declared In
`NSMenu.h`

contextMenuRepresentation

Deprecated. (Deprecated in Mac OS X v10.2. Mac OS X does not use menu representations to draw menus.)

```
- (id)contextMenuRepresentation
```

Return Value
`nil`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSMenu.h

delegate

Returns the receiver's delegate.

```
- (id < NSMenuDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setDelegate: (page 1629)

Declared In
NSMenu.h

font

Returns the font used to display the menu and its submenus.

```
- (NSFont *)font
```

Return Value
The font object used for displaying the menu.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setFont: (page 1630)

Declared In
NSMenu.h

helpRequested:

Overridden by subclasses to implement specialized context-sensitive help behavior.

```
- (void)helpRequested:(NSEvent *)event
```

Parameters
event
An NSEvent object representing the event associated with the help request.

Discussion
Subclasses in their implementation of this method should cause the Help Manager (NSHelpManager) to display the help associated with the receiver. Never invoke this method directly.

Special Considerations
On Mac OS X v10.6 and later this method has no effect. This method may be deprecated in a future release.

Availability
Available in Mac OS X v10.0 and later.

See Also
`showContextHelpForObject:locationHint:` (page 1320) (`NSHelpManager`)

Declared In
`NSMenu.h`

highlightedItem

Returns the highlighted item in the receiver.

```
- (NSMenuItem *)highlightedItem
```

Return Value
Returns the highlighted item in the receiver, or `nil` if no item in the menu is highlighted.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSMenu.h`

indexOfItem:

Returns the index identifying the location of a specified menu item in the receiver.

```
- (NSInteger)indexOfItem:(NSMenuItem *)anObject
```

Parameters
anObject
A menu item—that is an object conforming to the `NSMenuItem` protocol.

Return Value
The integer index of the menu item or, if no such menu item is in the menu, –1.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- insertItem:atIndex:` (page 1618)
`- itemAtIndex:` (page 1620)

Declared In
`NSMenu.h`

indexOfItemWithRepresentedObject:

Returns the index of the first menu item in the receiver that has a given represented object.

```
- (NSInteger)indexOfItemWithRepresentedObject:(id)anObject
```

Parameters

anObject

A represented object of the receiver.

Return Value

The integer index of the menu item or, if no such menu item is in the menu, –1.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `indexOfItemWithTag:` (page 1615)
- `indexOfItemWithTitle:` (page 1616)
- `insertItem:atIndex:` (page 1618)
- `itemAtIndex:` (page 1620)

Declared In

`NSMenu.h`

indexOfItemWithSubmenu:

Returns the index of the menu item in the receiver with the given submenu.

```
- (NSInteger)indexOfItemWithSubmenu:(NSMenu *)anObject
```

Parameters

anObject

A menu object that is a menu item of the receiver (that is, a submenu).

Return Value

The integer index of the menu item or, if no such menu item is in the menu, –1.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `insertItem:atIndex:` (page 1618)
- `itemAtIndex:` (page 1620)

Declared In

`NSMenu.h`

indexOfItemWithTag:

Returns the index of the first menu item in the receiver identified by a tag.

```
- (NSInteger)indexOfItemWithTag:(NSInteger)aTag
```

Parameters

aTag

An integer tag associated with the menu item of the receiver.

Return Value

The integer index of the menu item or, if no such menu item is in the menu, –1.

Availability

Available in Mac OS X v10.0 and later.

See Also

- itemWithTag: (page 1621)
- insertItem:atIndex: (page 1618)
- itemAtIndex: (page 1620)

Declared In

NSMenu.h

indexOfItemWithTarget:andAction:

Returns the index of the first menu item in the receiver that has a specified action and target.

```
- (NSInteger)indexOfItemWithTarget:(id)anObject andAction:(SEL)actionSelector
```

Parameters

anObject

An object that is set as the target of a menu item of the receiver.

actionSelector

A selector identifying an action method. If *actionSelector* is NULL, the first menu item in the receiver that has target *anObject* is returned

Return Value

The integer index of the menu item or, if no such menu item is in the menu, –1.

Availability

Available in Mac OS X v10.0 and later.

See Also

- indexOfItemWithTag: (page 1615)
- indexOfItemWithTitle: (page 1616)
- indexOfItemWithRepresentedObject: (page 1615)
- insertItem:atIndex: (page 1618)
- itemAtIndex: (page 1620)

Declared In

NSMenu.h

indexOfItemWithTitle:

Returns the index of the first menu item in the receiver that has a specified title.

```
- (NSInteger)indexOfItemWithTitle:(NSString *)aTitle
```

Parameters

aTitle

The title of a menu item in the receiver.

Return Value

The integer index of the menu item or, if no such menu item is in the menu, –1.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `itemWithTitle:` (page 1622)
- `insertItem:atIndex:` (page 1618)
- `itemAtIndex:` (page 1620)

Declared In

`NSMenu.h`

initWithTitle:

Initializes and returns a menu having the specified title and with autoenabling of menu items turned on.

```
- (id)initWithTitle:(NSString *)aTitle
```

Parameters

aTitle

The title to assign to the receiver.

Return Value

The initialized `NSMenu` object or `nil` if the object could not be initialized.

Special Considerations

This method is the designated initializer for the class.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setAutoenablesItems:` (page 1629)

Related Sample Code

DesktopImage
GLUT
MenuItemView
MenuMadness
SearchField

Declared In

`NSMenu.h`

insertItem:atIndex:

Inserts a menu item into the receiver at a specific location.

```
- (void)insertItem:(NSMenuItem *)newItem atIndex:(NSInteger)index
```

Parameters

newItem

An object conforming to the `NSMenuItem` protocol that represents a menu item.

index

An integer index identifying the location of the menu item in the menu.

Discussion

This method posts an `NSMenuDidAddItemNotification` (page 1637), allowing interested observers to update as appropriate. This method is a primitive method. All item-addition methods end up calling this method, so this is where you should implement custom behavior on adding new items to a menu in a custom subclass. If the menu item already exists in another menu, it is not inserted and the method raises an exception of type `NSInternalInconsistencyException`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addItem:` (page 1609)
- `itemAtIndex:` (page 1620)
- `removeItem:` (page 1627)

Related Sample Code

ButtonMadness

SearchField

Declared In

`NSMenu.h`

insertItemWithTitle:action:keyEquivalent:atIndex:

Creates and adds a menu item at a specified location in the receiver.

```
- (NSMenuItem *)insertItemWithTitle:(NSString *)aString action:(SEL)aSelector
    keyEquivalent:(NSString *)keyEquiv atIndex:(NSInteger)index
```

Parameters

aString

A string to be made the title of the menu item.

aSelector

The action-message selector to assign to the menu item.

keyEquiv

A string identifying the key to use as a key equivalent for the menu item. If you do not want the menu item to have a key equivalent, *keyEquiv* should be an empty string (@"") and not `nil`.

index

An integer index identifying the location of the menu item in the menu.

Return Value
The new menu item (an object conforming to the `NSMenuItem` protocol) or `nil` if the item could not be created

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ButtonMadness

Declared In
`NSMenu.h`

isAttached

Returns a Boolean value that indicates whether the receiver is currently attached to another menu. (Deprecated in Mac OS X v10.2.)

```
- (BOOL)isAttached
```

Return Value
`YES` if the receiver is currently attached to another menu, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

See Also
`- attachedMenu` (page 1611)

Declared In
`NSMenu.h`

isTornOff

Returns a Boolean value that indicates whether the receiver is offscreen or attached to another menu (or if it's the main menu).

```
- (BOOL)isTornOff
```

Return Value
`NO` if the receiver is offscreen or attached to another menu (or if it's the main menu), otherwise `YES`.

Special Considerations
On Mac OS X v10.6 and later this method has no effect.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMenu.h`

itemArray

Returns an array containing the receiver's menu items.

```
- (NSArray *)itemArray
```

Return Value
An array containing the receiver's menu items.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `itemWithTag:` (page 1621)
- `itemWithTitle:` (page 1622)
- `itemAtIndex:` (page 1620)
- `numberOfItems` (page 1624)

Related Sample Code
EnhancedAudioBurn
WhackedTV

Declared In
`NSMenu.h`

itemAtIndex:

Returns the menu item at a specific location of the receiver.

```
- (NSMenuItem *)itemAtIndex:(NSInteger)index
```

Parameters
index
An integer index locating a menu item in a menu.

Return Value
The found menu item (an object conforming to the `NSMenuItem` protocol) or `nil` if the object couldn't be found.

Discussion
This method raises an exception if *index* is out of bounds.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfItem:` (page 1614)
- `itemWithTag:` (page 1621)
- `itemWithTitle:` (page 1622)
- `itemArray` (page 1620)

Related Sample Code
DeskPictAppDockMenu
DragNDropOutlineView

MenuItemView
OutputBins2PDE
QTCoreVideo101

Declared In
NSMenu.h

itemChanged:

Invoked when a menu item is modified visually (for example, its title changes).

```
- (void)itemChanged:(NSMenuItem *)anObject
```

Parameters

anObject

The menu item that has visually changed.

Discussion
This method is not called for changes involving the menu item's action, target, represented object, or tag. Posts an `NSMenuDidChangeItemNotification` (page 1637).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSMenu.h

itemWithTag:

Returns the first menu item in the receiver with the specified tag.

```
- (NSMenuItem *)itemWithTag:(NSInteger)aTag
```

Parameters

aTag

A numeric tag associated with a menu item.

Return Value
The found menu item (an object conforming to the `NSMenuItem` protocol) or `nil` if the object couldn't be found.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfItemWithTag:` (page 1615)
- `itemWithTitle:` (page 1622)
- `itemAtIndex:` (page 1620)
- `itemArray` (page 1620)

Declared In
NSMenu.h

itemWithTitle:

Returns the first menu item in the receiver with a specified title.

```
- (NSMenuItem *)itemWithTitle:(NSString *)aString
```

Parameters

aString

The title of a menu item.

Return Value

The found menu item (an object conforming to the `NSMenuItem` protocol) or `nil` if the object couldn't be found.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `indexOfItemWithTitle:` (page 1616)
- `itemWithTag:` (page 1621)
- `itemAtIndex:` (page 1620)
- `itemArray` (page 1620)

Declared In

`NSMenu.h`

locationForSubmenu:

Returns the location in screen coordinates where the given submenu is displayed when opened as a submenu of the receiver. (Deprecated in Mac OS X v10.2.)

```
- (NSPoint)locationForSubmenu:(NSMenu *)aSubmenu
```

Parameters

aSubmenu

A menu object that is a submenu of the receiver.

Return Value

An `NSPoint` structure describing the location or (0.0, 0.0) if the submenu does not exist in the receiver.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.2.

Declared In

`NSMenu.h`

menuBarHeight

Returns the menu bar height for the current application's main menu.

```
- (CGFloat)menuBarHeight
```

Return Value
The receiver's main menu bar height or 0.0 if the receiver is some other menu.

Discussion
This method supersedes the `menuBarHeight` (page 1688) class method of the `NSMenuView` class.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSMenu.h`

menuChangedMessagesEnabled

Returns a Boolean value that indicates whether messages are sent to the application's windows upon each change to the receiver.

```
- (BOOL)menuChangedMessagesEnabled
```

Return Value
`YES` if messages are sent to the application's windows upon each change to the receiver, otherwise `NO`.

Special Considerations

On Mac OS X v10.6 and later this method has no effect. This method may be deprecated in a future release.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMenuChangedMessagesEnabled:` (page 1630)

Declared In
`NSMenu.h`

menuRepresentation

Deprecated. (Deprecated in Mac OS X v10.2. Mac OS X does not use menu representations to draw menus.)

```
- (id)menuRepresentation
```

Return Value
`nil`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
`NSMenu.h`

minimumWidth

Returns the minimum width of the menu.

```
- (CGFloat)minimumWidth
```

Return Value
The minimum width of the menu in screen coordinates.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSMenu.h`

numberOfItems

Returns the number of menu items in the receiver, including separator items.

```
- (NSInteger)numberOfItems
```

Return Value
The number of menu items in the receiver, including separator items.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- itemArray` (page 1620)

Related Sample Code
DeskPictAppDockMenu
QTCoreVideo101

Declared In
`NSMenu.h`

performActionForItemAtIndex:

Causes the application to send the action message of a specified menu item to its target.

```
- (void)performActionForItemAtIndex:(NSInteger)index
```

Parameters
index
The integer index of a menu item.

Discussion
If a target is not specified, the message is sent to the first responder. As a side effect, this method posts `NSMenuWillSendActionNotification` (page 1639) and `NSMenuDidSendActionNotification` (page 1638).

In Mac OS X v10.6 and later the `performActionForItemAtIndex:` no longer triggers menu validation. This is because validation is typically done during menu tracking or key equivalent matching, so the subsequent `performActionForItemAtIndex:` validation was redundant. To trigger validation explicitly, use invoke the `update` (page 1635) method.

In Mac OS X v10.6 `performActionForItemAtIndex:`, when called, now triggers highlighting in the menu bar. It also sends out appropriate accessibility notifications indicating the item was selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- performKeyEquivalent:` (page 1625)

Declared In
`NSMenu.h`

performKeyEquivalent:

Performs the action for the menu item that corresponds to the given key equivalent.

```
- (BOOL)performKeyEquivalent:(NSEvent *)theEvent
```

Parameters

theEvent
An `NSEvent` object that represents a key-equivalent event.

Return Value
`YES` if *theEvent* is a key equivalent that the receiver handled, `NO` if it is not a key equivalent that it should handle.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- performActionForItemAtIndex:` (page 1624)
`menuHasKeyEquivalent:forEvent:target:action:` (page 3726) (NSMenuDelegate)

Declared In
`NSMenu.h`

popUpMenuPositioningItem:atLocation:inView:

Pops up the menu at the specified location.

```
- (BOOL)popUpMenuPositioningItem:(NSMenuItem *)item atLocation:(NSPoint)location
    inView:(NSView *)view
```

Parameters

item
The menu item to be positioned at the specified location in the view.

location
The location in the *view* coordinate system to display the menu item.

view
The view to display the menu item over.

Return Value
YES if menu tracking ended because an item was selected, and NO if menu tracking was cancelled for any reason.

Discussion
Pops up the receiver as a popup menu. The top left corner of the specified item (if specified, *item* must be present in the receiver) is positioned at the specified location in the specified view, interpreted in the view's own coordinate system.

If *item* is nil, the menu is positioned such that the top left of the menu content frame is at the given location.

If *view* is nil, the location is interpreted in the screen coordinate system. This allows you to pop up a menu disconnected from any window.

Availability
Available in Mac OS X v10.6 and later.

See Also
+ popUpContextMenu:withEvent:forView: (page 1607)
+ popUpContextMenu:withEvent:forView:withFont: (page 1608)

Declared In
NSMenu.h

propertiesToUpdate

Returns the available properties for the menu.

```
- (NSMenuProperties)propertiesToUpdate
```

Return Value
A bitwise-C OR of the values in the "NSMenuProperties" (page 1636) that are applicable to this menu.

Discussion
The propertiesToUpdate method may be called on the menu from specific callbacks to determine which properties the have been defined, and whether or not they are relevant to the changes you need to make. It is intended to allow more efficient updating of the menu in certain circumstances.

For example, if NSMenuPropertyItemImage (page 1636) is not set, your delegate does not need to update the images of the menu items, because the images are not needed (for example, during key equivalent matching). If the NSMenuPropertyItemImage (page 1636) bit is 0, you can avoid updating the menu's images, which may improve performance if computing the images is expensive.

You only have to update a property if it may have changed since you last set it, even if the corresponding bit is 1. For example, if the title of a menu item never changes, you only have to set it once.

You may call this from the menu delegate methods menuNeedsUpdate: (page 3727), or the menu validation methods -validateMenuItem: (page 989) or validateUserInterfaceItem: (page 3923). Calling this at other times will raise an exception.

If a menu property does not have a corresponding bit, you should ensure it is always set properly after the callback returns.

Calling this is optional; it is always acceptable to fully update all properties of the menu.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSMenu.h

removeAllItems

Removes all the menu items in the receiver.

```
- (void)removeAllItems
```

Discussion
This method is more efficient than removing menu items individually.

Unlike the other remove methods, this method does not post `NSMenuDidChangeItemNotification` (page 1637) notifications.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `removeItem:` (page 1627)
- `removeItemAtIndex:` (page 1628)

Declared In
NSMenu.h

removeItem:

Removes a menu item from the receiver.

```
- (void)removeItem:(NSMenuItem *)anItem
```

Parameters
anItem
The menu item to remove.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addItem:` (page 1609)
- `addItemWithTitle:action:keyEquivalent:` (page 1610)
- `removeItemAtIndex:` (page 1628)

Declared In
NSMenu.h

removeItemAtIndex:

Removes the menu item at a specified location in the receiver.

```
- (void)removeItemAtIndex:(NSInteger)index
```

Parameters

index

An integer index identifying the menu item.

Discussion

After it removes the menu item, this method posts an `NSMenuDidRemoveItemNotification` (page 1638).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addItem:` (page 1609)
- `addItemWithTitle:action:keyEquivalent:` (page 1610)
- `removeItem:` (page 1627)

Related Sample Code

ButtonMadness
DeskPictAppDockMenu
ObjectPath
QTAudioContextInsert
QTAudioExtractionPanel

Declared In

`NSMenu.h`

setAllowsContextMenuPlugIns:

Sets whether the popup menu allows appending of contextual menu plugin items.

```
- (void)setAllowsContextMenuPlugIns:(BOOL)allows
```

Parameters

allows

`YES` if the popup menu should allow context menu plugin items to be appending, otherwise `NO`.

Discussion

Contextual menu plugins are system-wide services provided by other applications. For example, a contextual menu plugin might provide an "Open URL..." service. By enabling context menu plugins your application's contextual menu will display the appropriate items for the currently selected data type.

See *Services Implementation Guide* for more information on contextual menu plugins.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSMenu.h`

setAutoenablesItems:

Controls whether the receiver automatically enables and disables its menu items based on delegates implementing the `NSMenuValidation` informal protocol.

```
- (void)setAutoenablesItems:(BOOL)flag
```

Parameters

flag

If *flag* is `YES`, menu items are automatically enabled and disabled. If *flag* is `NO`, menu items are not automatically enabled or disabled.

Discussion
See the `NSMenuValidation` protocol specification for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- autoenablesItems` (page 1611)

Related Sample Code
GLUT
SearchField

Declared In
`NSMenu.h`

setContextMenuRepresentation:

Deprecated. (Deprecated in Mac OS X v10.2. Mac OS X does not use menu representations to draw menus.)

```
- (void)setContextMenuRepresentation:(id)menuRep
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
`NSMenu.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSMenuDelegate >)anObject
```

Parameters

anObject

The object to set as delegate.

Discussion
You can use the delegate to populate a menu just before it is going to be drawn and to check for key equivalents without creating a menu item.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `delegate` (page 1613)

Declared In
`NSMenu.h`

setFont:

Sets the font used to display the menu and its submenus.

```
- (void)setFont:(NSFont *)font
```

Parameters

font

The font object to use.

Discussion
This font will be used to display the menu and any submenus that have not had their font set explicitly.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `font` (page 1613)

Declared In
`NSMenu.h`

setMenuChangedMessagesEnabled:

Controls whether the receiver sends messages to the application's windows upon each menu change.

```
- (void)setMenuChangedMessagesEnabled:(BOOL)flag
```

Parameters

flag

`YES` if the receiver should send a message at each menu change, `NO` otherwise.

Discussion
To avoid the "flickering" effect of many successive menu changes, invoke this method with *flag* set to `NO`, make changes to the menu, and invoke the method again with *flag* set to `YES`. This approach has the effect of batching changes and applying them all at once.

Special Considerations

On Mac OS X v10.6 and later this method has no effect. This method may be deprecated in a future release.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `menuChangedMessagesEnabled` (page 1623)

Declared In
NSMenu.h

setMenuRepresentation:

Deprecated. (Deprecated in Mac OS X v10.2. Mac OS X does not use menu representations to draw menus.)

```
- (void)setMenuRepresentation:(id)menuRep
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSMenu.h

setMinimumWidth:

Set the minimum width of the menu.

```
- (void)setMinimumWidth:(CGFloat)width
```

Parameters
width
The minimum width of the menu in screen coordinates.

Discussion
The menu will not draw smaller than its minimum width, but may draw larger if it needs more space. The default value is 0.

Availability
Available in Mac OS X v10.6 and later.

See Also
- minimumWidth (page 1624)

Declared In
NSMenu.h

setShowsStateColumn:

Sets whether the receiver displays the state column.

```
- (void)setShowsStateColumn:(BOOL)showsState
```

Parameters
showsState
YES to display the state column, otherwise NO.

Availability
Available in Mac OS X v10.5 and later.

See Also
- showsStateColumn (page 1633)

Declared In
NSMenu.h

setSubmenu:forItem:

Assigns a menu to be a submenu of the receiver controlled by a given menu item.

```
- (void)setSubmenu:(NSMenu *)aMenu forItem:(NSMenuItem *)anItem
```

Parameters

aMenu

A menu object that is to be a submenu of the receiver.

anItem

A menu item (that is, an object conforming to the NSMenuItem protocol) that controls *aMenu*. The method sets the action of *anItem* to submenuAction: (page 1634).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSMenu.h

setSupermenu:

Sets the receiver's supermenu.

```
- (void)setSupermenu:(NSMenu *)supermenu
```

Parameters

supermenu

A menu object to set as the supermenu of the receiver.

Discussion
You should never invoke this method directly; it is public so subclassers can add behavior to the default implementation. Subclassers should call the superclass's method as part of their implementation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- supermenu (page 1634)

Declared In
NSMenu.h

setTearOffMenuRepresentation:

Deprecated. (Deprecated in Mac OS X v10.2. Mac OS X does not use menu representations to draw menus.)

```
- (void)setTearOffMenuRepresentation:(id)menuRep
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSMenu.h

setTitle:

Sets the receiver's title.

```
- (void)setTitle:(NSString *)aString
```

Parameters
aString
A string to assign as the new title of the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- title (page 1635)

Declared In
NSMenu.h

showsStateColumn

Returns a Boolean value that indicates whether the receiver displays the state column.

```
- (BOOL)showsStateColumn
```

Return Value
YES if the receiver displays the state column, otherwise NO.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setShowsStateColumn: (page 1631)

Declared In
NSMenu.h

size

Returns the size of the menu.

```
- (NSSize)size
```

Return Value
The size of the menu in screen coordinates.

Discussion
The menu may draw at a smaller size when shown, depending on its positioning and display configuration.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSMenu.h`

sizeToFit

Resizes the receiver to exactly fit its items. (Deprecated in Mac OS X v10.2.)

```
- (void)sizeToFit
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
`NSMenu.h`

submenuAction:

The action method assigned to menu items that open submenus.

```
- (void)submenuAction:(id)sender
```

Discussion
You may override this method to implement different behavior. Never invoke this method directly.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMenu.h`

supermenu

Returns the receiver's supermenu.

```
- (NSMenu *)supermenu
```

Return Value
The receiver's supermenu or `nil` if it has none.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setSupermenu: (page 1632)

Declared In
NSMenu.h

tearOffMenuRepresentation

Deprecated. (Deprecated in Mac OS X v10.2. Mac OS X does not use menu representations to draw menus.)

```
- (id)tearOffMenuRepresentation
```

Return Value
nil.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSMenu.h

title

Returns the receiver's title.

```
- (NSString *)title
```

Return Value
The receiver's title.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTitle: (page 1633)

Related Sample Code
ToolbarSample

Declared In
NSMenu.h

update

Enables or disables the receiver's menu items based on the NSMenuValidation informal protocol and sizes the menu to fit its current menu items if necessary.

```
- (void)update
```

Discussion
See the NSMenuValidation protocol specification for more information.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSMenu.h

Constants

NSMenuProperties

These constants are used as a bitmask for specifying a set of menu or menu item properties, used in the propertiesToUpdate (page 1626).

```
enum {
    NSMenuPropertyItemTitle = 1 << 0,
    NSMenuPropertyItemAttributedTitle = 1 << 1
    NSMenuPropertyItemKeyEquivalent = 1 << 2,
    NSMenuPropertyItemImage = 1 << 3,
    NSMenuPropertyItemEnabled = 1 << 4,
    NSMenuPropertyItemAccessibilityDescription = 1 << 5
};
typedef NSUInteger NSMenuProperties;
```

Constants

NSMenuPropertyItemTitle
The menu item's title.
Available in Mac OS X v10.6 and later.
Declared in NSMenu.h.

NSMenuPropertyItemAttributedTitle
The menu item's attributed string title.
Available in Mac OS X v10.6 and later.
Declared in NSMenu.h.

NSMenuPropertyItemKeyEquivalent
The menu item's key equivalent.
Available in Mac OS X v10.6 and later.
Declared in NSMenu.h.

NSMenuPropertyItemImage
The menu image.
Available in Mac OS X v10.6 and later.
Declared in NSMenu.h.

NSMenuPropertyItemEnabled
Whether the menu item is enabled or disabled.
Available in Mac OS X v10.6 and later.
Declared in NSMenu.h.

`NSMenuPropertyItemAccessibilityDescription`
The menu item's accessibility description.

Available in Mac OS X v10.6 and later.

Declared in `NSMenu.h`.

Notifications

NSMenuDidAddItemNotification

Posted after a menu item is added to the menu. The notification object is the instance of `NSMenu` that just added the new menu item. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSMenuItemIndex"`	An `NSNumber` object containing the integer index of the menu item that was added.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMenu.h`

NSMenuDidChangeItemNotification

Posted after a menu item in the menu changes appearance. Changes include enabling/disabling, changes in state, and changes to title. The notification object is the instance of `NSMenu` with the menu item that changed. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSMenuItemIndex"`	An `NSNumber` object containing the integer index of the menu item that changed.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMenu.h`

NSMenuDidBeginTrackingNotification

Posted when menu tracking begins. The notification object is the main menu bar (`[NSApp mainMenu]`) or the root menu of a popup button. This notification does not contain a *`userInfo`* dictionary.

Note: This notification is available in versions 10.3 and 10.4 of Mac OS X, however it is not publicly declared so you must declare the name constant as an `extern`, for example:

```
extern NSString *NSMenuDidBeginTrackingNotification;
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSMenu.h`

NSMenuDidEndTrackingNotification

Posted when menu tracking ends, even if no action is sent. The notification object is the main menu bar (`[NSApp mainMenu]`) or the root menu of a popup button. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSMenu.h`

NSMenuDidRemoveItemNotification

Posted after a menu item is removed from the menu. The notification object is the instance of `NSMenu` that just removed the menu item. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSMenuItemIndex"`	An `NSNumber` object containing the integer index of the menu item that was removed. Note that this index may no longer be valid and in any event no longer points to the menu item that was removed.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMenu.h`

NSMenuDidSendActionNotification

Posted just after the application dispatches a menu item's action method to the menu item's target. The notification object is the instance of `NSMenu` containing the chosen menu item. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"MenuItem"`	The menu item that was chosen.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMenu.h`

NSMenuWillSendActionNotification

Posted just before the application dispatches a menu item's action method to the menu item's target. The notification object is the instance of `NSMenu` containing the chosen menu item. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"MenuItem"`	The menu item that was chosen.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMenu.h`

NSMenuItem Class Reference

Inherits from	NSObject
Conforms to	NSValidatedUserInterfaceItem NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSMenuItem.h
Companion guide	Application Menu and Pop-up List Programming Topics
Related sample code	MenuMadness PDFKitLinker2 QTAudioContextInsert Quartz Composer WWDC 2005 TextEdit SearchField

Overview

The `NSMenuItem` class defines objects that are used as command items in menus. Additionally, the `NSMenuItem` class also includes some private functionality needed to maintain binary compatibility with other components of Cocoa. Because of this fact, you cannot replace the `NSMenuItem` class with a different class. You may, however, subclass `NSMenuItem` if necessary.

Prior to Mac OS X v10.5, `NSMenuItem` conformed to the following protocols: `NSCopying` (see *NSCopying Protocol Reference*), `NSCoding` (see *NSCoding Protocol Reference*), and `NSValidatedUserInterfaceItem` (see *NSValidatedUserInterfaceItem Protocol Reference*).

Tasks

Creating a Menu Item

`- initWithTitle:action:keyEquivalent:` (page 1649)
Returns an initialized instance of an `NSMenuItem`.

Enabling a Menu Item

- `setEnabled:` (page 1658)
  Sets whether the receiver is enabled
- `isEnabled` (page 1650)
  Returns a Boolean value that indicates whether the receiver is enabled.

Managing Hidden Status

- `setHidden:` (page 1659)
  Sets whether the receiver is hidden.
- `isHidden` (page 1650)
  Returns a Boolean value that indicates whether the receiver is hidden.
- `isHiddenOrHasHiddenAncestor` (page 1651)
  Returns a Boolean value that indicates whether the receiver or any of its superitems is hidden.

Managing the Target and Action

- `setTarget:` (page 1666)
  Sets the receiver's target.
- `target` (page 1670)
  Returns the receiver's target.
- `setAction:` (page 1656)
  Sets the receiver's action-method selector.
- `action` (page 1647)
  Returns the receiver's action-method selector.

Managing the Title

- `setTitle:` (page 1667)
  Sets the receiver's title.
- `title` (page 1671)
  Returns the receiver's title.
- `setAttributedTitle:` (page 1657)
  Specifies a custom string for a menu item.
- `attributedTitle` (page 1648)
  Returns the custom title string for a menu item.

Managing the Tag

- `setTag:` (page 1666)
  Sets the receiver's tag.

- `tag` (page 1670)
  Returns the receiver's tag.

Managing the State

- `setState:` (page 1665)
  Sets the state of the receiver.
- `state` (page 1669)
  Returns the state of the receiver.

Managing the Image

- `setImage:` (page 1659)
  Sets the receiver's image.
- `image` (page 1648)
  Returns the image displayed by the receiver.
- `setOnStateImage:` (page 1663)
  Sets the image of the receiver that indicates an "on" state.
- `onStateImage` (page 1655)
  Returns the image used to depict the receiver's "on" state.
- `setOffStateImage:` (page 1663)
  Sets the image of the receiver that indicates an "off" state.
- `offStateImage` (page 1654)
  Returns the image used to depict the receiver's "off" state.
- `setMixedStateImage:` (page 1662)
  Sets the image of the receiver that indicates a "mixed" state, that is, a state neither "on" nor "off."
- `mixedStateImage` (page 1653)
  Returns the image used to depict a "mixed state."

Managing Submenus

- `setSubmenu:` (page 1665)
  Sets the submenu of the receiver.
- `submenu` (page 1669)
  Returns the submenu associated with the receiving menu item.
- `hasSubmenu` (page 1648)
  Returns a Boolean value that indicates whether the receiver has a submenu.
- `parentItem` (page 1655)
  Returns the menu item whose submenu contains the receiver.

Getting a Separator Item

+ `separatorItem` (page 1646)
  Returns a menu item that is used to separate logical groups of menu commands.

- `isSeparatorItem` (page 1651)
  Returns a Boolean value that indicates whether the receiver is a separator item.

Managing the Owning Menu

- `setMenu:` (page 1661)
  Sets the receiver's menu.

- `menu` (page 1653)
  Returns the menu to which the receiver belongs.

Managing Key Equivalents

- `setKeyEquivalent:` (page 1660)
  Sets the receiver's unmodified key equivalent.

- `keyEquivalent` (page 1652)
  Returns the receiver's unmodified keyboard equivalent.

- `setKeyEquivalentModifierMask:` (page 1661)
  Sets the receiver's keyboard equivalent modifiers.

- `keyEquivalentModifierMask` (page 1652)
  Returns the receiver's keyboard equivalent modifier mask.

Managing Mnemonics

- `setTitleWithMnemonic:` (page 1667)
  Deprecated. Sets the title of a menu item with a character denoting an access key.

- `mnemonic` (page 1653) Deprecated in Mac OS X v10.6
  Deprecated. Returns the character in the menu item title that appears underlined for use as a mnemonic.

- `mnemonicLocation` (page 1654) Deprecated in Mac OS X v10.6
  Deprecated. Returns the position of the underlined character in the menu item title used as a mnemonic.

- `setMnemonicLocation:` (page 1662) Deprecated in Mac OS X v10.6
  Deprecated. Sets the character of the menu item title at location that is to be underlined.

Managing User Key Equivalents

+ `setUsesUserKeyEquivalents:` (page 1646)
  Sets whether menu items conform to user preferences for key equivalents.

+ `usesUserKeyEquivalents` (page 1647)
Returns a Boolean value that indicates whether menu items conform to user preferences for key equivalents.

- `userKeyEquivalent` (page 1671)
Returns the user-assigned key equivalent for the receiver.

Managing Alternates

- `setAlternate:` (page 1656)
Marks the receiver as an alternate to the previous menu item.

- `isAlternate` (page 1650)
Returns a Boolean value that indicates whether the receiver is an alternate to the previous menu item.

Managing Indentation Levels

- `setIndentationLevel:` (page 1660)
Sets the menu item indentation level for the receiver.

- `indentationLevel` (page 1649)
Returns the menu item indentation level for the receiver.

Managing Tool Tips

- `setToolTip:` (page 1668)
Sets a help tag for a menu item.

- `toolTip` (page 1671)
Returns the help tag for a menu item.

Representing an Object

- `setRepresentedObject:` (page 1664)
Sets the object represented by the receiver.

- `representedObject` (page 1655)
Returns the object that the receiving menu item represents.

Managing the View

- `setView:` (page 1668)
Sets the content view for the receiver.

- `view` (page 1672)
Returns the view for the receiver.

Getting Highlighted Status

`- isHighlighted` (page 1651)
Returns a Boolean value that indicates whether the receiver should be drawn highlighted.

Class Methods

separatorItem

Returns a menu item that is used to separate logical groups of menu commands.

```
+ (NSMenuItem *)separatorItem
```

Return Value
A menu item that is used to separate logical groups of menu commands.

Discussion
This menu item is disabled. The default separator item is blank space.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- isSeparatorItem` (page 1651)
`- setEnabled:` (page 1658)

Related Sample Code
DesktopImage
MenuMadness
QTAudioContextInsert
QTAudioExtractionPanel
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSMenuItem.h`

setUsesUserKeyEquivalents:

Sets whether menu items conform to user preferences for key equivalents.

```
+ (void)setUsesUserKeyEquivalents:(BOOL)flag
```

Parameters
flag
If `YES`, menu items conform to user preferences for key equivalents; otherwise, the key equivalents originally assigned to the menu items are used.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ usesUserKeyEquivalents (page 1647)
- userKeyEquivalent (page 1671)

Declared In
NSMenuItem.h

usesUserKeyEquivalents

Returns a Boolean value that indicates whether menu items conform to user preferences for key equivalents.

```
+ (BOOL)usesUserKeyEquivalents
```

Return Value
YES if menu items conform to user preferences for key equivalents, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ setUsesUserKeyEquivalents: (page 1646)
- userKeyEquivalent (page 1671)

Declared In
NSMenuItem.h

Instance Methods

action

Returns the receiver's action-method selector.

```
- (SEL)action
```

Return Value
The receiver's action-method selector.

Availability
Available in Mac OS X v10.0 and later.

See Also
- target (page 1670)
- setAction: (page 1656)

Related Sample Code
EnhancedAudioBurn
QTKitPlayer
Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility
Sketch-112

Declared In
NSMenuItem.h

attributedTitle

Returns the custom title string for a menu item.

```
- (NSAttributedString *)attributedTitle
```

Return Value
The custom title string for a menu item.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setAttributedTitle: (page 1657)
- title (page 1671)

Declared In
NSMenuItem.h

hasSubmenu

Returns a Boolean value that indicates whether the receiver has a submenu.

```
- (BOOL)hasSubmenu
```

Return Value
YES if the receiver has a submenu, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setSubmenu:forItem: (page 1632) (NSMenu)

Declared In
NSMenuItem.h

image

Returns the image displayed by the receiver.

```
- (NSImage *)image
```

Return Value
The image displayed by the receiver, or nil if it displays no image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setImage:` (page 1659)

Declared In
`NSMenuItem.h`

indentationLevel

Returns the menu item indentation level for the receiver.

```
- (NSInteger)indentationLevel
```

Discussion
The return value is from 0 to 15. The default indentation level is 0.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setIndentationLevel:` (page 1660)

Declared In
`NSMenuItem.h`

initWithTitle:action:keyEquivalent:

Returns an initialized instance of an `NSMenuItem`.

```
- (id)initWithTitle:(NSString *)itemName
    action:(SEL)anAction
    keyEquivalent:(NSString *)charCode
```

Parameters

itemName
The title of the menu item. This value must not be `nil` (if there is no title, specify an empty `NSString`).

anAction
The action selector to be associated with the menu item. This value must be a valid selector or `NULL`.

charCode
A string representing a keyboard key to be used as the key equivalent. This value must not be `nil` (if there is no key equivalent, specify an empty `NSString`).

Return Value
An instance of `NSMenuItem`, or `nil` if the object couldn't be created.

Discussion
For instances of the `NSMenuItem` class, the default initial state is `NSOffState`, the default on-state image is a check mark, and the default mixed-state image is a dash.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DesktopImage

MenuItemView
MenuMadness
PDFKitLinker2
SearchField

Declared In
NSMenuItem.h

isAlternate

Returns a Boolean value that indicates whether the receiver is an alternate to the previous menu item.

```
- (BOOL)isAlternate
```

Return Value
YES if the receiver is an alternate to the previous menu item, otherwise NO.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setAlternate: (page 1656)

Declared In
NSMenuItem.h

isEnabled

Returns a Boolean value that indicates whether the receiver is enabled.

```
- (BOOL)isEnabled
```

Return Value
YES if the receiver is enabled, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setEnabled: (page 1658)

Related Sample Code
QTAudioContextInsert

Declared In
NSMenuItem.h

isHidden

Returns a Boolean value that indicates whether the receiver is hidden.

```
- (BOOL)isHidden
```

Return Value
`YES` if the receiver is hidden, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setHidden:` (page 1659)
- `isHiddenOrHasHiddenAncestor` (page 1651)

Declared In
`NSMenuItem.h`

isHiddenOrHasHiddenAncestor

Returns a Boolean value that indicates whether the receiver or any of its superitems is hidden.

```
- (BOOL)isHiddenOrHasHiddenAncestor
```

Return Value
`YES` if the receiver or any of its superitems is hidden, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setHidden:` (page 1659)
- `isHidden` (page 1650)

Declared In
`NSMenuItem.h`

isHighlighted

Returns a Boolean value that indicates whether the receiver should be drawn highlighted.

```
- (BOOL)isHighlighted
```

Return Value
`YES` if the receiver should be drawn highlighted, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSMenuItem.h`

isSeparatorItem

Returns a Boolean value that indicates whether the receiver is a separator item.

```
- (BOOL)isSeparatorItem
```

Return Value
`YES` if the receiver is a separator item (that is, a menu item used to visually segregate related menu items), otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMenuItem.h`

keyEquivalent

Returns the receiver's unmodified keyboard equivalent.

```
- (NSString *)keyEquivalent
```

Return Value
The receiver's unmodified keyboard equivalent, or the empty string if one hasn't been defined.

Discussion
Use `keyEquivalentModifierMask` (page 1652) to determine the modifier mask for the key equivalent.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `userKeyEquivalent` (page 1671)
- `mnemonic` (page 1653)
- `setKeyEquivalent:` (page 1660)

Declared In
`NSMenuItem.h`

keyEquivalentModifierMask

Returns the receiver's keyboard equivalent modifier mask.

```
- (NSUInteger)keyEquivalentModifierMask
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setKeyEquivalentModifierMask:` (page 1661)

Declared In
`NSMenuItem.h`

menu

Returns the menu to which the receiver belongs.

```
- (NSMenu *)menu
```

Return Value
The menu to which the receiver belongs, or `nil` if no menu has been set.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMenu:` (page 1661)

Related Sample Code
WhackedTV

Declared In
`NSMenuItem.h`

mixedStateImage

Returns the image used to depict a “mixed state.”

```
- (NSImage *)mixedStateImage
```

Return Value
The image used to depict a “mixed state.”

Discussion
A mixed state is useful for indicating a mix of “off” and “on” attribute values in a group of selected objects, such as a selection of text containing boldface and plain (non-boldface) words. By default this is a horizontal line.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMixedStateImage:` (page 1662)

Declared In
`NSMenuItem.h`

mnemonic

Deprecated. Returns the character in the menu item title that appears underlined for use as a mnemonic. (Deprecated in Mac OS X v10.6.)

```
- (NSString *)mnemonic
```

Discussion
If there is no mnemonic character, returns an empty string.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `setTitleWithMnemonic:` (page 1667)

Declared In
`NSMenuItem.h`

mnemonicLocation

Deprecated. Returns the position of the underlined character in the menu item title used as a mnemonic. (Deprecated in Mac OS X v10.6.)

```
- (NSUInteger)mnemonicLocation
```

Discussion
The position is the zero-based index of that character in the title string. If the receiver has no mnemonic character, returns `NSNotFound`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `setMnemonicLocation:` (page 1662)

Declared In
`NSMenuItem.h`

offStateImage

Returns the image used to depict the receiver's "off" state.

```
- (NSImage *)offStateImage
```

Return Value
The image used to depict the receiver's "off" state, or `nil` if the image has not been set.

Discussion
By default there is no off-state image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setOffStateImage:` (page 1663)

Declared In
`NSMenuItem.h`

onStateImage

Returns the image used to depict the receiver's "on" state.

```
- (NSImage *)onStateImage
```

Return Value
The image used to depict the receiver's "on" state, or `nil` if the image has not been set.

Discussion
By default this image is a check mark.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setOnStateImage:` (page 1663)

Declared In
`NSMenuItem.h`

parentItem

Returns the menu item whose submenu contains the receiver.

```
- (NSMenuItem *)parentItem
```

Return Value
The parent menu item, or `nil` if the receiver does not have a parent item.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSMenuItem.h`

representedObject

Returns the object that the receiving menu item represents.

```
- (id)representedObject
```

Discussion
For example, you might have a menu list the names of views that are swapped into the same panel. The represented objects would be the appropriate `NSView` objects. The user would then be able to switch back and forth between the different views that are displayed by selecting the various menu items.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `tag` (page 1670)
- `setRepresentedObject:` (page 1664)

Related Sample Code
DesktopImage
OutputBins2PDE
QTAudioContextInsert
QTAudioExtractionPanel
UIElementInspector

Declared In
NSMenuItem.h

setAction:

Sets the receiver's action-method selector.

```
- (void)setAction:(SEL)aSelector
```

Parameters
aSelector
A selector identifying the action method.

Discussion
See *Action Messages* for additional information on action messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTarget: (page 1666)
- action (page 1647)

Related Sample Code
MenuMadness
NumberInput_IMKit_Sample
Quartz Composer WWDC 2005 TextEdit
UIElementInspector

Declared In
NSMenuItem.h

setAlternate:

Marks the receiver as an alternate to the previous menu item.

```
- (void)setAlternate:(BOOL)isAlternate
```

Parameters
isAlternate
YES if the receiver is an alternate to the previous menu item, NO otherwise.

Discussion
If the receiver has the same key equivalent as the previous item, but has different key equivalent modifiers, the items are folded into a single visible item and the appropriate item shows while tracking the menu, depending on what modifier key (if any) is pressed. The menu items may also have no key equivalent as long as the key equivalent modifiers are different.

Consider the following example: `menuItem1` and `menuItem2` are menu items in the same menu, with `menuItem1` above `menuItem2`:

```
[menuItem1 setTitle:@"One"];
[menuItem1 setKeyEquivalent:@"t"];

[menuItem2 setTitle:@"Two"];
[menuItem2 setKeyEquivalent:@"T"];
[menuItem2 setAlternate:YES];
```

When the menu is displayed, it shows only `menuItem1` (with title "One") instead of two menu items. If the user presses the Shift key while the menu is displayed, `menuItem2` (with title "Two") replaces "One".

If there are two or more items with no key equivalent but different modifiers, then the only way to get access to the alternate items is with the mouse. In the following example,"Two" is shown only if the user presses the Alternate key.

```
[menuItem1 setKeyEquivalent:@""];
[menuItem1 setTitle:@"One"];

[menuItem2 setKeyEquivalent:@""];
[menuItem2 setKeyEquivalentModifierMask:NSAlternateKeyMask];
[menuItem2 setTitle:@"Two"];
```

If you mark items as alternates but their key equivalents don't match, they might be displayed as separate items. Marking the first item as an alternate has no effect.

The *`isAlternate`* value is archived.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- isAlternate` (page 1650)

Declared In
`NSMenuItem.h`

setAttributedTitle:

Specifies a custom string for a menu item.

```
- (void)setAttributedTitle:(NSAttributedString *)string
```

Parameters
`string`
An attributed string to use as the receiver's title.

Discussion
You can use this method to add styled text and embedded images to menu item strings. If you do not set a text color for the attributed string, it is black when not selected, white when selected, and gray when disabled. Colored text remains unchanged when selected.

When you call this method to set the menu title to an attributed string, the `setTitle:` (page 1667) method is also called to set the menu title with a plain string. If you clear the attributed title, the plain title remains unchanged. To clear the attributed title, set the attributed string to either `nil` or an empty attributed string (`[attrStr length] == 0`).

The attributed string is not archived in the old nib format.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `attributedTitle` (page 1648)
- `setTitle:` (page 1667)

Declared In
`NSMenuItem.h`

setEnabled:

Sets whether the receiver is enabled

```
- (void)setEnabled:(BOOL)flag
```

Parameters
flag
`YES` if the receiver is to be enabled, otherwise `NO`.

Discussion
This method has no effect unless the menu in which the item will be added or is already a part of has been sent `setAutoenablesItems:NO`. If a menu item is disabled, its keyboard equivalent is also disabled. See the `NSMenuValidation` informal protocol specification for cautions regarding this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isEnabled` (page 1650)

Related Sample Code
DeskPictAppDockMenu
DragNDropOutlineView
MenuItemView
QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSMenuItem.h`

setHidden:

Sets whether the receiver is hidden.

```
- (void)setHidden:(BOOL)hidden
```

Parameters

hidden

`YES` if the receiver is to be hidden, otherwise `NO`.

Discussion

Hidden menu items (or items with a hidden superitem) do not appear in a menu and do not participate in command key matching.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `isHidden` (page 1650)
- `isHiddenOrHasHiddenAncestor` (page 1651)

Declared In

`NSMenuItem.h`

setImage:

Sets the receiver's image.

```
- (void)setImage:(NSImage *)menuImage
```

Parameters

menuImage

An `NSImage` object representing an image to be displayed in the menu item. If *menuImage* is `nil`, the current image (if any) is removed.

Discussion

The menu item's image is not affected by changes in its state.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `image` (page 1648)

Related Sample Code

ButtonMadness

DesktopImage

MenuItemView

MenuMadness

SourceView

Declared In

`NSMenuItem.h`

setIndentationLevel:

Sets the menu item indentation level for the receiver.

```
- (void)setIndentationLevel:(NSInteger)indentationLevel
```

Parameters

indentationLevel

The value for *indentationLevel* may be from 0 to 15. If *indentationLevel* is greater than 15, the value is pinned to the maximum. If *indentationLevel* is less than 0, an exception is raised. The default indentation level is 0.

Discussion

The *indentationLevel* value is archived.

Availability

Available in Mac OS X v10.3 and later.

See Also

- indentationLevel (page 1649)

Declared In

NSMenuItem.h

setKeyEquivalent:

Sets the receiver's unmodified key equivalent.

```
- (void)setKeyEquivalent:(NSString *)aString
```

Parameters

aString

A string containing a character code representing a keyboard key. If you want to remove the key equivalent from a menu item, pass an empty string (@"") for *aString* (never pass nil).

Discussion

This method considers the case of the letter passed to determine if it has a Shift modifier added. That is, [item setKeyEquivalent:@"w"] sets the key equivalent to Command-w, while [item setKeyEquivalent:@"W"] is Command-Shift-w. You use setKeyEquivalentModifierMask: (page 1661) to set the appropriate mask for the modifier keys for the key equivalent.

If you want to specify the Backspace key as the key equivalent for a menu item, use a single character string with NSBackspaceCharacter (page 2752) (defined in NSText.h as 0x08) and for the Forward Delete key, use NSDeleteCharacter (page 2752) (defined in NSText.h as 0x7F). Note that these are not the same characters you get from an NSEvent key-down event when pressing those keys.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setMnemonicLocation: (page 1662)
- keyEquivalent (page 1652)

Related Sample Code

CocoaDVDPlayer

Declared In
NSMenuItem.h

setKeyEquivalentModifierMask:

Sets the receiver's keyboard equivalent modifiers.

```
- (void)setKeyEquivalentModifierMask:(NSUInteger)mask
```

Parameters

mask

The key masks indicate modifiers such as the Shift or Option keys. *mask* is an integer bit field containing any of these modifier key masks, combined using the C bitwise OR operator:

NSShiftKeyMask
NSAlternateKeyMask
NSCommandKeyMask
NSControlKeyMask

Discussion
In general, you are strongly encouraged to always set NSCommandKeyMask in *mask*, although there may be some conventions where this is not required. For example, in an application that plays media, the Play command may be mapped to just " " (space), without the command key. You can do this with the following code:

```
[menuItem setKeyEquivalent:@" "];
[menuItem setKeyEquivalentModifierMask:0];
```

NSShiftKeyMask is a valid modifier for any key equivalent in *mask*. This allows you to specify key-equivalents such as Command-Shift-1 that are consistent across all keyboards. However, with a few exceptions (such as the German "ß" character), a lowercase character with NSShiftKeyMask is interpreted the same as the uppercase character without that mask. For example, Command-Shift-c and Command-C are considered to be identical key equivalents.

See the NSEvent class specification for more information about modifier mask values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- keyEquivalentModifierMask (page 1652)

Related Sample Code
MenuMadness

Declared In
NSMenuItem.h

setMenu:

Sets the receiver's menu.

```
- (void)setMenu:(NSMenu *)aMenu
```

Parameters

aMenu

The menu object that "owns" the receiver.

Discussion

This method is invoked by the owning `NSMenu` object when the receiver is added or removed. You shouldn't have to invoke this method in your own code, although it can be overridden to provide specialized behavior.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `menu` (page 1653)

Declared In

`NSMenuItem.h`

setMixedStateImage:

Sets the image of the receiver that indicates a "mixed" state, that is, a state neither "on" nor "off."

```
- (void)setMixedStateImage:(NSImage *)itemImage
```

Parameters

itemImage

The `NSImage` object to use for the "mixed" state of the menu item. If *itemImage* is `nil`, any current mixed-state image is removed.

Discussion

Changing state images is currently unsupported in Mac OS X.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `mixedStateImage` (page 1653)
- `setOffStateImage:` (page 1663)
- `setOnStateImage:` (page 1663)
- `setState:` (page 1665)

Related Sample Code

MenuItemView

Declared In

`NSMenuItem.h`

setMnemonicLocation:

Deprecated. Sets the character of the menu item title at location that is to be underlined. (Deprecated in Mac OS X v10.6.)

```
- (void)setMnemonicLocation:(NSUInteger)location
```

Parameters

location

An integer index into the character array of the title. *location* must be from 0 to 254.

Discussion

This character identifies the access key by which users can access the menu item.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- mnemonicLocation (page 1654)

Declared In

NSMenuItem.h

setOffStateImage:

Sets the image of the receiver that indicates an “off” state.

```
- (void)setOffStateImage:(NSImage *)itemImage
```

Parameters

itemImage

The NSImage object to use for the "off" state of the menu item. If *itemImage* is nil, any current off-state image is removed.

Discussion

Changing state images is currently unsupported in Mac OS X.

Availability

Available in Mac OS X v10.0 and later.

See Also

- offStateImage (page 1654)
- setMixedStateImage: (page 1662)
- setOffStateImage: (page 1663)
- setState: (page 1665)

Declared In

NSMenuItem.h

setOnStateImage:

Sets the image of the receiver that indicates an “on” state.

```
- (void)setOnStateImage:(NSImage *)itemImage
```

Parameters

itemImage

The NSImage object to use for the "on" state of the menu item. If *itemImage* is nil, any current on-state image is removed.

Discussion
Changing state images is currently unsupported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `onStateImage` (page 1655)
- `setMixedStateImage:` (page 1662)
- `setOffStateImage:` (page 1663)
- `setState:` (page 1665)

Related Sample Code
MenuItemView
MenuMadness

Declared In
`NSMenuItem.h`

setRepresentedObject:

Sets the object represented by the receiver.

```
- (void)setRepresentedObject:(id)anObject
```

Parameters
anObject
The object to be represented by the receiver.

Discussion
By setting a represented object for a menu item, you make an association between the menu item and that object. The represented object functions as a more specific form of tag that allows you to associate any object, not just an arbitrary integer, with the items in a menu.

For example, an `NSView` object might be associated with a menu item—when the user chooses the menu item, the represented object is fetched and displayed in a panel. Several menu items might control the display of multiple views in the same panel.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTag:` (page 1666)
- `representedObject` (page 1655)

Related Sample Code
DeskPictAppDockMenu
DesktopImage
QTAudioContextInsert
QTAudioExtractionPanel
UIElementInspector

Declared In
NSMenuItem.h

setState:

Sets the state of the receiver.

```
- (void)setState:(NSInteger)itemState
```

Parameters

itemState

An integer constant representing a state; it should be one of NSOffState, NSOnState, or NSMixedState.

Discussion
The image associated with the new state is displayed to the left of the menu item.

Availability
Available in Mac OS X v10.0 and later.

See Also
- state (page 1669)
- setMixedStateImage: (page 1662)
- setOffStateImage: (page 1663)
- setOnStateImage: (page 1663)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer
Sketch-112

Declared In
NSMenuItem.h

setSubmenu:

Sets the submenu of the receiver.

```
- (void)setSubmenu:(NSMenu *)aSubmenu
```

Parameters

aSubmenu

The menu object to set as submenu.

Discussion
The default implementation of the NSMenuItem class raises an exception if *aSubmenu* already has a supermenu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `submenu` (page 1669)
- `hasSubmenu` (page 1648)

Related Sample Code
MenuItemView
MenuMadness
PDFKitLinker2
ToolbarSample
UIElementInspector

Declared In
`NSMenuItem.h`

setTag:

Sets the receiver's tag.

```
- (void)setTag:(NSInteger)anInt
```

Parameters

anInt
An integer tag to associate with the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setRepresentedObject:` (page 1664)
- `tag` (page 1670)

Related Sample Code
MenuMadness
QTAudioContextInsert
QTAudioExtractionPanel
Quartz Composer WWDC 2005 TextEdit
SearchField

Declared In
`NSMenuItem.h`

setTarget:

Sets the receiver's target.

```
- (void)setTarget:(id)anObject
```

Parameters

anObject
An object to be the target of action messages sent by the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAction:` (page 1656)
- `target` (page 1670)

Related Sample Code
MenuItemView
MenuMadness
PDFKitLinker2
Quartz Composer WWDC 2005 TextEdit
ScannerBrowser

Declared In
`NSMenuItem.h`

setTitle:

Sets the receiver's title.

```
- (void)setTitle:(NSString *)aString
```

Parameters
aString
The new title of the menu item. If you do not want a title, use an empty string (@"") , not `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `title` (page 1671)
- `setAttributedTitle:` (page 1657)

Related Sample Code
CocoaDVDPlayer
ColorMatching
FunHouse
QTAudioExtractionPanel
Sketch+Accessibility

Declared In
`NSMenuItem.h`

setTitleWithMnemonic:

Deprecated. Sets the title of a menu item with a character denoting an access key.

```
- (void)setTitleWithMnemonic:(NSString *)aString
```

Discussion
Use an ampersand character to mark the character (the one following the ampersand) to be designated.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `mnemonic` (page 1653)
- `setMnemonicLocation:` (page 1662)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSMenuItem.h`

setToolTip:

Sets a help tag for a menu item.

```
- (void)setToolTip:(NSString *)toolTip
```

Parameters
toolTip
A short string that describes the menu item.

Discussion
You can invoke this method for any menu item, including items in the main menu bar. This string is not archived in the old nib format.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `toolTip` (page 1671)

Declared In
`NSMenuItem.h`

setView:

Sets the content view for the receiver.

```
- (void)setView:(NSView *)view
```

Parameters
view
The content view for the receiver.

Discussion
A menu item with a view does not draw its title, state, font, or other standard drawing attributes, and assigns drawing responsibility entirely to the view. Keyboard equivalents and type-select continue to use the key equivalent and title as normal. For more details, see *Application Menu and Pop-up List Programming Topics*.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `view` (page 1672)

Related Sample Code
MenuItemView

Declared In
`NSMenuItem.h`

state

Returns the state of the receiver.

```
- (NSInteger)state
```

Return Value
The state of the receiver—one of `NSOffState` (the default), `NSOnState`, or `NSMixedState`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setState:` (page 1665)

Declared In
`NSMenuItem.h`

submenu

Returns the submenu associated with the receiving menu item.

```
- (NSMenu *)submenu
```

Return Value
The submenu associated with the receiving menu item, or `nil` if no submenu is associated with it.

Discussion
If the receiver responds `YES` to `hasSubmenu` (page 1648), the submenu is returned.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hasSubmenu` (page 1648)
- `setSubmenu:` (page 1665)

Related Sample Code
EnhancedAudioBurn

Declared In
NSMenuItem.h

tag

Returns the receiver's tag.

```
- (NSInteger)tag
```

Return Value
The receiver's tag.

Availability
Available in Mac OS X v10.0 and later.

See Also
- representedObject (page 1655)
- setTag: (page 1666)

Related Sample Code
QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit
ThreadsExporter
ThreadsImporter
ThreadsImportMovie

Declared In
NSMenuItem.h

target

Returns the receiver's target.

```
- (id)target
```

Return Value
The receiver's target.

Availability
Available in Mac OS X v10.0 and later.

See Also
- action (page 1647)
- setTarget: (page 1666)

Related Sample Code
DeskPictAppDockMenu
EnhancedDataBurn

Declared In
NSMenuItem.h

title

Returns the receiver's title.

```
- (NSString *)title
```

Return Value
The receiver's title.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTitle:` (page 1667)

Related Sample Code
EnhancedAudioBurn
WhackedTV

Declared In
`NSMenuItem.h`

toolTip

Returns the help tag for a menu item.

```
- (NSString *)toolTip
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setToolTip:` (page 1668)

Declared In
`NSMenuItem.h`

userKeyEquivalent

Returns the user-assigned key equivalent for the receiver.

```
- (NSString *)userKeyEquivalent
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `keyEquivalent` (page 1652)

Declared In
`NSMenuItem.h`

view

Returns the view for the receiver.

```
- (NSView *)view
```

Return Value
The view for the receiver.

Discussion
By default, a menu item has a `nil` view.

See `setView:` (page 1668) for more details.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setView:` (page 1668)

Declared In
`NSMenuItem.h`

NSMenuItemCell Class Reference

Inherits from	NSButtonCell : NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSMenuItemCell.h
Companion guide	Application Menu and Pop-up List Programming Topics

Overview

`NSMenuItemCell` is a class that handles the measurement and display of a single menu item in its encompassing frame. Instances of `NSMenuItemCell` work in conjunction with an `NSMenuView` object to control the overall appearance of the menu.

Note: `NSMenuItemCell` is no longer used to draw menus. Using it will not affect the appearance of your menus.

Tasks

Configuring Menu-Item Attributes

- `menuItem` (page 1679)
  Returns the `NSMenuItem` object associated with the receiver.
- `setMenuItem:` (page 1681)
  Sets the `NSMenuItem` object associated withthe receive.
- `menuView` (page 1680)
  Returns the menu view associated with the receiver.
- `setMenuView:` (page 1681)
  Sets the menu view for the receiver.

Calculating the Size of a Menu Item

- `- calcSize` (page 1675)
  Calculates the minimum required width and height of the receiver's menu item.
- `- needsSizing` (page 1680)
  Returns `YES` if the size of the menu item needs to be calculated; otherwise returns `NO`.
- `- setNeedsSizing:` (page 1682)
  Sets a flag that indicates whether or not the menu item must be resized.
- `- imageWidth` (page 1678)
  Returns the width of the image associated with a menu item.
- `- titleWidth` (page 1684)
  Returns the width of the menu item text.
- `- keyEquivalentWidth` (page 1679)
  Returns the width of the key equivalent associated with the menu item.
- `- stateImageWidth` (page 1683)
  Returns the width of the image used to indicate the state of the menu item.

Getting the Menu Item's Drawing Rectangle

- `- keyEquivalentRectForBounds:` (page 1679)
  Returns the rectangle into which the menu item's key equivalent should be drawn.
- `- stateImageRectForBounds:` (page 1682)
  Returns the rectangle into which the menu item's state image should be drawn.
- `- titleRectForBounds:` (page 1683)
  Returns the rectangle into which the menu item's title should be drawn.

Drawing the Menu Item

- `- drawBorderAndBackgroundWithFrame:inView:` (page 1675)
  Draws the borders and background associated with the receiver's menu item (if any).
- `- drawImageWithFrame:inView:` (page 1676)
  Draws the image associated with the menu item.
- `- drawKeyEquivalentWithFrame:inView:` (page 1676)
  Draws the key equivalent associated with the menu item.
- `- drawSeparatorItemWithFrame:inView:` (page 1677)
  Draws a menu item separator.
- `- drawStateImageWithFrame:inView:` (page 1677)
  Draws the state image associated with the menu item.
- `- drawTitleWithFrame:inView:` (page 1678)
  Draws the title associated with the menu item.
- `- needsDisplay` (page 1680)
  Returns `YES` if the menu item needs to be displayed; otherwise returns `NO`.

- `setNeedsDisplay:` (page 1681)
  Sets whether the menu item needs to be drawn.

Assigning a Tag

- `tag` (page 1683)
  Returns the integer tag of the selected menu item, or 0 if no item is selected.

Instance Methods

calcSize

Calculates the minimum required width and height of the receiver's menu item.

```
- (void)calcSize
```

Discussion
The calculated values are cached for future use. This method also calculates the sizes of individual components of the cell's menu item and caches those values.

This method is invoked automatically when necessary. You should not need to invoke it directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `needsSizing` (page 1680)

Declared In
`NSMenuItemCell.h`

drawBorderAndBackgroundWithFrame:inView:

Draws the borders and background associated with the receiver's menu item (if any).

```
- (void)drawBorderAndBackgroundWithFrame:(NSRect)cellFrame inView:(NSView
    *)controlView
```

Parameters
cellFrame
  A rectangle defining the receiver's frame area.
controlView
  The view object that contains this cell (usually an `NSControl` object).

Discussion
This method invokes the `NSCell` method `imageRectForBounds:` (page 563), passing it *cellFrame*, to calculate the rectangle in which to draw the image. The cell invokes this method before invoking the methods to draw the other menu item components.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- drawWithFrame:inView:` (page 554) (NSCell)

Declared In
`NSMenuItemCell.h`

drawImageWithFrame:inView:

Draws the image associated with the menu item.

```
- (void)drawImageWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
```

Parameters

cellFrame
A rectangle defining the receiver's frame area.

controlView
The view object that contains this cell (usually an `NSControl` object).

Discussion
This method invokes the `NSCell` method `imageRectForBounds:` (page 563), passing it *cellFrame*, to calculate the rectangle in which to draw the image. This method is invoked by the cell's `drawWithFrame:` method. You should not need to invoke it directly. Subclasses may override this method to control the drawing of the image.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSMenuItemCell.h`

drawKeyEquivalentWithFrame:inView:

Draws the key equivalent associated with the menu item.

```
- (void)drawKeyEquivalentWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
```

Parameters

cellFrame
A rectangle defining the receiver's frame area.

controlView
The view object that contains this cell (usually an `NSControl` object).

Discussion
This method invokes `keyEquivalentRectForBounds:` (page 1679), passing it *cellFrame*, to calculate the rectangle in which to draw the key equivalent. This method is invoked by the cell's `drawWithFrame:` method. You should not need to invoke it directly. Subclasses may override this method to control the drawing of the key equivalent.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSMenuItemCell.h

drawSeparatorItemWithFrame:inView:

Draws a menu item separator.

```
- (void)drawSeparatorItemWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
```

Parameters

cellFrame
A rectangle defining the receiver's frame area.

controlView
The view object that contains this cell (usually an NSControl object).

Discussion
This method uses the *cellFrame* parameter to calculate the rectangle in which to draw the menu item separator. This method uses the *controlView* to determine whether the separator item should be drawn normally or flipped.

You should not need to invoke this method directly. Subclasses may override this method to control the drawing of the separator.

Availability
Available in Mac OS X v10.0 and later.

See Also
- drawKeyEquivalentWithFrame:inView: (page 1676)
- drawTitleWithFrame:inView: (page 1678)
- isFlipped (page 3181) (NSView)

Declared In
NSMenuItemCell.h

drawStateImageWithFrame:inView:

Draws the state image associated with the menu item.

```
- (void)drawStateImageWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
```

Parameters

cellFrame
A rectangle defining the receiver's frame area.

controlView
The view object that contains this cell (usually an NSControl object).

Discussion
This method invokes stateImageRectForBounds: (page 1682), passing it *cellFrame*, to calculate the rectangle in which to draw the state image. This method is invoked by the cell's drawWithFrame: method. You should not need to invoke it directly. Subclasses may override this method to control the drawing of the state image.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSMenuItemCell.h

drawTitleWithFrame:inView:

Draws the title associated with the menu item.

```
- (void)drawTitleWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
```

Parameters

cellFrame
A rectangle defining the receiver's frame area.

controlView
The view object that contains this cell (usually an `NSControl` object).

Discussion
This method invokes `titleRectForBounds:` (page 1683), passing it *cellFrame*, to calculate the rectangle in which to draw the title. The *controlView* parameter specifies the view that contains this cell.This method is invoked by the cell's `drawWithFrame:` method. You should not need to invoke it directly. Subclasses may override this method to control the drawing of the title.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSMenuItemCell.h

imageWidth

Returns the width of the image associated with a menu item.

```
- (CGFloat)imageWidth
```

Discussion
You can associate an image with a menu item using the `NSMenuItem setImage:` method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `stateImageWidth` (page 1683)
- `calcSize` (page 1675)
- `needsSizing` (page 1680)

Declared In
NSMenuItemCell.h

keyEquivalentRectForBounds:

Returns the rectangle into which the menu item's key equivalent should be drawn.

```
- (NSRect)keyEquivalentRectForBounds:(NSRect)cellFrame
```

Parameters

cellFrame

A rectangle that defines the bounds of the receiver.

Return Value

The returned rectangle is based on *cellFrame* but encompasses only the area to be occupied by the key equivalent.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `keyEquivalent` (NSMenuItem)
- `stateImageRectForBounds:` (page 1682)
- `titleRectForBounds:` (page 1683)
- `keyEquivalentRectForBounds:` (page 1679)

Declared In

`NSMenuItemCell.h`

keyEquivalentWidth

Returns the width of the key equivalent associated with the menu item.

```
- (CGFloat)keyEquivalentWidth
```

Discussion

You can associate a key equivalent with a menu item using the `NSMenuItem` method `setKeyEquivalent:`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `calcSize` (page 1675)
- `needsSizing` (page 1680)

Declared In

`NSMenuItemCell.h`

menuItem

Returns the `NSMenuItem` object associated with the receiver.

```
- (NSMenuItem *)menuItem
```

Availability

Available in Mac OS X v10.0 and later.

See Also
- `setMenuItem:` (page 1681)

Declared In
`NSMenuItemCell.h`

menuView

Returns the menu view associated with the receiver.

```
- (NSMenuView *)menuView
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `setMenuView:` (page 1681)

Declared In
`NSMenuItemCell.h`

needsDisplay

Returns `YES` if the menu item needs to be displayed; otherwise returns `NO`.

```
- (BOOL)needsDisplay
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setNeedsDisplay:` (page 1681)

Declared In
`NSMenuItemCell.h`

needsSizing

Returns `YES` if the size of the menu item needs to be calculated; otherwise returns `NO`.

```
- (BOOL)needsSizing
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setNeedsSizing:` (page 1682)
- `calcSize` (page 1675)

Declared In
NSMenuItemCell.h

setMenuItem:

Sets the NSMenuItem object associated withthe receive.

```
- (void)setMenuItem:(NSMenuItem *)item
```

Parameters

item

The NSMenuItem object to set.

Availability
Available in Mac OS X v10.0 and later.

See Also
- menuItem (page 1679)

Declared In
NSMenuItemCell.h

setMenuView:

Sets the menu view for the receiver.

```
- (void)setMenuView:(NSMenuView *)menuView
```

Parameters

menuView

The NSMenuView object to associate with the receiver.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- menuView (page 1680)

Declared In
NSMenuItemCell.h

setNeedsDisplay:

Sets whether the menu item needs to be drawn.

```
- (void)setNeedsDisplay:(BOOL)flag
```

Parameters

flag

YES if the menu item needs to be drawn, NO otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- needsDisplay (page 1680)

Declared In
NSMenuItemCell.h

setNeedsSizing:

Sets a flag that indicates whether or not the menu item must be resized.

```
- (void)setNeedsSizing:(BOOL)flag
```

Parameters

flag

If *flag* is YES, the next attempt to obtain any size-related information from this menu item cell invokes the calcSize (page 1675) method to recalculate the information. If *flag* is NO, the next attempt to obtain size-related information returns the currently cached values.

Discussion
Subclasses that drastically change the way a menu item is drawn may need to invoke this method to recalculate the menu item information. Other parts of your application should not need to invoke this method directly. The cell invokes this method as necessary when the content of its menu item changes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- needsSizing (page 1680)

Declared In
NSMenuItemCell.h

stateImageRectForBounds:

Returns the rectangle into which the menu item's state image should be drawn.

```
- (NSRect)stateImageRectForBounds:(NSRect)cellFrame
```

Parameters

cellFrame

A rectangle that defines the bounds of the receiver.

Return Value
The returned rectangle is based on *cellFrame* but encompasses only the area to be occupied by the menu item's state image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- stateImageRectForBounds: (page 1682)

- titleRectForBounds: (page 1683)
- keyEquivalentRectForBounds: (page 1679)

Declared In
NSMenuItemCell.h

stateImageWidth

Returns the width of the image used to indicate the state of the menu item.

```
- (CGFloat)stateImageWidth
```

Discussion
If the menu item has multiple images associated with it (to indicate any of the available states: on, off, or mixed), this method returns the width of the largest image. You can set the state images for a menu item using the NSMenuItem methods setOnStateImage:, setOffStateImage:, and setMixedStateImage:.

To change the state of the cell's menu item, use the NSMenuItem method setState:.

Availability
Available in Mac OS X v10.0 and later.

See Also
- calcSize (page 1675)
- needsSizing (page 1680)
- setState: (NSMenuItem)

Declared In
NSMenuItemCell.h

tag

Returns the integer tag of the selected menu item, or 0 if no item is selected.

```
- (NSInteger)tag
```

Discussion
Setting the tag value of an NSMenuItemCell object with setTag: (page 69) does nothing.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setTag: (page 69) (NSActionCell)

Declared In
NSMenuItemCell.h

titleRectForBounds:

Returns the rectangle into which the menu item's title should be drawn.

```
- (NSRect)titleRectForBounds:(NSRect)cellFrame
```

Parameters

cellFrame

A rectangle that defines the bounds of the receiver.

Return Value

The returned rectangle is based on *cellFrame* but encompasses only the area to be occupied by the text of the title.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `stateImageRectForBounds:` (page 1682)
- `titleRectForBounds:` (page 1683)
- `keyEquivalentRectForBounds:` (page 1679)

Declared In

`NSMenuItemCell.h`

titleWidth

Returns the width of the menu item text.

```
- (CGFloat)titleWidth
```

Discussion

To set the menu item text, use NSMenuItem's `setTitle:` method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `calcSize` (page 1675)
- `needsSizing` (page 1680)

Declared In

`NSMenuItemCell.h`

NSMenuView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later. Not available to 64-bit applications.
Declared in	AppKit/NSMenuView.h
Companion guide	Application Menu and Pop-up List Programming Topics

Overview

The NSMenuView class handles the display of menus on the user's screen. A menu view displays its menu either horizontally or vertically and allows the user to interact with the items of that menu, either to navigate through hierarchical menus or to select a particular item.

Note: `NSMenuView` is deprecated and is no longer used to draw menus. Calling its methods will not affect the appearance of your menus. Instead, use `NSView` customizations on an `NSMenuItem`.

Tasks

Initializing a Menu View

`- initAsTearOff` (page 1692)
Deprecated. Tear-off menus are not supported in Mac OS X.

`- initWithFrame:` (page 1693)
Initialized a newly allocated menu view with a specified frame rectangle.

Managing Menu View Attributes

+ `menuBarHeight` (page 1688)
  Returns the height of the menu bar.

- `setMenu:` (page 1700)
  Sets the menu to be displayed in the receiver

- `menu` (page 1697)
  Returns the menu object associated with this menu view.

- `setHorizontal:` (page 1700)
  Sets the orientation of the menu.

- `isHorizontal` (page 1694)
  Returns `YES` if the menu is displayed horizontally; such as for a menu bar, otherwise returns `NO`.

- `setFont:` (page 1699)
  Sets the default font to use when drawing the menu text.

- `font` (page 1690)
  Returns the default font used to draw the menu text.

- `setHighlightedItemIndex:` (page 1699)
  Highlights the menu item at a specific location.

- `highlightedItemIndex` (page 1690)
  Returns the index of the currently highlighted menu item, or –1 if no menu item in the menu is highlighted.

- `setMenuItemCell:forItemAtIndex:` (page 1701)
  Replaces the menu item cell at a specific location.

- `menuItemCellForItemAtIndex:` (page 1697)
  Returns the menu item cell at the specified location.

- `attachedMenuView` (page 1689)
  Returns the receiver's attached menu view.

- `isTornOff` (page 1694)
  Deprecated. Tear-off menus are not supported in Mac OS X.

- `horizontalEdgePadding` (page 1691)
  Returns the amount of horizontal space used for padding menu item components.

- `setHorizontalEdgePadding:` (page 1700)
  Sets the horizontal padding for menu item components.

- `attachedMenu` (page 1688) Deprecated in Mac OS X v10.2
  Returns the menu object associated with this object's attached menu view.

- `isAttached` (page 1693) Deprecated in Mac OS X v10.2
  Returns `YES` if this menu is currently attached to its parent menu, `NO` otherwise.

Responding to Notifications

- `itemChanged:` (page 1695)
  Marks the menu view as needing to be resized so changes in size resulting from a change in the menu will be tracked.

- `itemAdded:` (page 1694)
  Creates a new menu item cell for the newly created item and marks the menu view as needing to be resized.
- `itemRemoved:` (page 1695)
  Removes the removed item's menu item cell and marks the menu view as needing to be resized.

Working With Submenus

- `detachSubmenu` (page 1690)
  Detaches the window associated with the currently visible submenu and removes any menu item highlights.
- `attachSubmenuForItemAtIndex:` (page 1689)
  Attaches the submenu associated with the menu item at *index*.

Calculating Menu Geometry

- `update` (page 1704)
  Asks the associated menu object to update itself.
- `setNeedsSizing:` (page 1702)
  Sets a flag that indicates whether the layout is invalid and needs resizing.
- `needsSizing` (page 1698)
  Returns `YES` if the menu view needs to be resized due to changes in the menu object, `NO` otherwise.
- `sizeToFit` (page 1703)
  Used internally by the menu view to cache information about the menu item geometry.
- `stateImageOffset` (page 1703)
  Returns the offset to the space reserved for state images of this menu.
- `stateImageWidth` (page 1703)
  Returns the maximum width of the state images used by this menu.
- `imageAndTitleOffset` (page 1691)
  Returns the offset to the starting point of a menu item's image and title section.
- `imageAndTitleWidth` (page 1691)
  Returns the maximum width of a menu item's image and title section.
- `keyEquivalentOffset` (page 1696)
  Returns the beginning position of the menu's key equivalent text.
- `keyEquivalentWidth` (page 1696)
  Returns the width of the menu's key equivalent text.
- `innerRect` (page 1693)
  Returns the drawing rectangle for the menu contents.
- `rectOfItemAtIndex:` (page 1698)
  Returns the drawing rectangle of the specified menu item.
- `indexOfItemAtPoint:` (page 1692)
  Returns the index of the menu item underneath the specified or –1 if no menu item is underneath that point.

- `setNeedsDisplayForItemAtIndex:` (page 1701)
  Adds the region occupied by the menu item at a specific location to the menu view's invalid region.
- `setWindowFrameForAttachingToRect:onScreen:preferredEdge:popUpSelectedItem:` (page 1702)
  Causes the menu view to resize its window so its frame is the appropriate size to attach to a specified rectangle within the screen.
- `locationForSubmenu:` (page 1696) Deprecated in Mac OS X v10.2
  Returns the origin of the submenu view's window.

Handling Events

- `performActionWithHighlightingForItemAtIndex:` (page 1698)
  Uses the associated menu object to perform the action associated with the specified item when a key equivalent is pressed.
- `trackWithEvent:` (page 1704)
  Handles events sent to this menu view.

Class Methods

menuBarHeight

Returns the height of the menu bar.

```
+ (CGFloat)menuBarHeight
```

Discussion
This method is superseded in Mac OS X v10.4 by the NSMenu `menuBarHeight` (page 1622) instance method.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`NSMenuView.h`

Instance Methods

attachedMenu

Returns the menu object associated with this object's attached menu view.

```
- (NSMenu *)attachedMenu
```

Discussion
The attached menu view is the one associated with the currently visible submenu, if any.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `attachedMenuView` (page 1689)
- `isAttached` (page 1693)

Declared In
`NSMenuView.h`

attachedMenuView

Returns the receiver's attached menu view.

```
- (NSMenuView *)attachedMenuView
```

Discussion
The attached menu view is the one associated with the currently visible submenu, if any.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `attachedMenu` (page 1688)
- `detachSubmenu` (page 1690)
- `isAttached` (page 1693)

Declared In
`NSMenuView.h`

attachSubmenuForItemAtIndex:

Attaches the submenu associated with the menu item at *index*.

```
- (void)attachSubmenuForItemAtIndex:(NSInteger)index
```

Discussion
This method prepares the submenu for display by positioning its window and ordering it to the front.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `setWindowFrameForAttachingToRect:onScreen:preferredEdge:popUpSelectedItem:` (page 1702)
- `orderFront:` (page 3351) (NSWindow)

Declared In
`NSMenuView.h`

detachSubmenu

Detaches the window associated with the currently visible submenu and removes any menu item highlights.

```
- (void)detachSubmenu
```

Discussion
If the submenu itself displays further submenus, this method detaches the windows associated with those submenus as well.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `attachSubmenuForItemAtIndex:` (page 1689)
- `setHighlightedItemIndex:` (page 1699)
- `orderOut:` (page 3352) (NSWindow)

Declared In
`NSMenuView.h`

font

Returns the default font used to draw the menu text.

```
- (NSFont *)font
```

Discussion
New items use this font by default, although the item's menu item cell can use a different font.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `setFont:` (page 1699)

Declared In
`NSMenuView.h`

highlightedItemIndex

Returns the index of the currently highlighted menu item, or –1 if no menu item in the menu is highlighted.

```
- (NSInteger)highlightedItemIndex
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `setHighlightedItemIndex:` (page 1699)

Declared In
NSMenuView.h

horizontalEdgePadding

Returns the amount of horizontal space used for padding menu item components.

```
- (CGFloat)horizontalEdgePadding
```

Discussion
The edge padding is added to the sides of each menu item component. This space is used to provide a visual separation between components of the menu item.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `setHorizontalEdgePadding:` (page 1700)

Declared In
NSMenuView.h

imageAndTitleOffset

Returns the offset to the starting point of a menu item's image and title section.

```
- (CGFloat)imageAndTitleOffset
```

Discussion
The image and title section of a menu item displays an image, a title, or possibly both as a way to identify the purpose of the menu item. The value returned by this method is used for all menu items of the menu.

If any changes have been made to the menu's contents, this method invokes `sizeToFit` (page 1703) to update the menu view information.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `imageAndTitleWidth` (page 1691)
- `stateImageOffset` (page 1703)
- `keyEquivalentOffset` (page 1696)

Declared In
NSMenuView.h

imageAndTitleWidth

Returns the maximum width of a menu item's image and title section.

```
- (CGFloat)imageAndTitleWidth
```

Discussion
The image and title section of a menu item displays an image, a title, or possibly both as a way to identify the purpose of the menu item. The value returned by this method is used for all menu items of the menu.

If any changes have been made to the menu's contents, this method invokes `sizeToFit` (page 1703) to update the menu view information.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `imageAndTitleOffset` (page 1691)
- `stateImageWidth` (page 1703)
- `keyEquivalentWidth` (page 1696)

Declared In
`NSMenuView.h`

indexOfItemAtPoint:

Returns the index of the menu item underneath the specified or –1 if no menu item is underneath that point.

```
- (NSInteger)indexOfItemAtPoint:(NSPoint)point
```

Discussion
This method considers the menu borders as part of the item when calculating whether *point* is in the menu item rectangle. This method invokes the `rectOfItemAtIndex:` (page 1698) method to obtain the basic rectangle for each menu item but may adjust that rectangle before testing.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `rectOfItemAtIndex:` (page 1698)

Declared In
`NSMenuView.h`

initAsTearOff

Deprecated. Tear-off menus are not supported in Mac OS X.

```
- (id)initAsTearOff
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
NSMenuView.h

initWithFrame:

Initialized a newly allocated menu view with a specified frame rectangle.

```
- (id)initWithFrame:(NSRect)frame
```

Discussion
This method is the designated initialization method for NSMenuView.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
NSMenuView.h

innerRect

Returns the drawing rectangle for the menu contents.

```
- (NSRect)innerRect
```

Discussion
This rectangle is different (typically smaller) from the view bounds in that it does not include the space used to draw the menu borders.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- bounds (page 3150) (NSView)

Declared In
NSMenuView.h

isAttached

Returns YES if this menu is currently attached to its parent menu, NO otherwise.

```
- (BOOL)isAttached
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- attachedMenu (page 1688)

- attachedMenuView (page 1689)

Declared In
NSMenuView.h

isHorizontal

Returns YES if the menu is displayed horizontally; such as for a menu bar, otherwise returns NO.

```
- (BOOL)isHorizontal
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- setHorizontal: (page 1700)

Declared In
NSMenuView.h

isTornOff

Deprecated. Tear-off menus are not supported in Mac OS X.

```
- (BOOL)isTornOff
```

Discussion
Returns YES if this menu view's window is disassociated from its parent menu.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
NSMenuView.h

itemAdded:

Creates a new menu item cell for the newly created item and marks the menu view as needing to be resized.

```
- (void)itemAdded:(NSNotification *)notification
```

Discussion
This method is registered with the menu view's associated NSMenu object for notifications of the type NSMenuDidAddItemNotification (page 1637). The *notification* parameter contains the notification data.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `setNeedsSizing:` (page 1702)

Declared In
`NSMenuView.h`

itemChanged:

Marks the menu view as needing to be resized so changes in size resulting from a change in the menu will be tracked.

```
- (void)itemChanged:(NSNotification *)notification
```

Discussion
This method is registered with the menu view's associated NSMenu object for notifications of the type `NSMenuDidChangeItemNotification` (page 1637). The *notification* parameter contains the notification data.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `setNeedsSizing:` (page 1702)

Declared In
`NSMenuView.h`

itemRemoved:

Removes the removed item's menu item cell and marks the menu view as needing to be resized.

```
- (void)itemRemoved:(NSNotification *)notification
```

Discussion
This method is registered with the menu view's associated NSMenu object for notifications of the type `NSMenuDidRemoveItemNotification` (page 1638). The *notification* parameter contains the notification data.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `setNeedsSizing:` (page 1702)

Declared In
`NSMenuView.h`

keyEquivalentOffset

Returns the beginning position of the menu's key equivalent text.

```
- (CGFloat)keyEquivalentOffset
```

Discussion
If any changes have been made to the menu's contents, this method invokes `sizeToFit` (page 1703) to update the menu view information.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `keyEquivalentWidth` (page 1696)
- `stateImageOffset` (page 1703)
- `imageAndTitleOffset` (page 1691)

Declared In
`NSMenuView.h`

keyEquivalentWidth

Returns the width of the menu's key equivalent text.

```
- (CGFloat)keyEquivalentWidth
```

Discussion
If any changes have been made to the menu's contents, this method invokes `sizeToFit` (page 1703) to update the menu view information.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `keyEquivalentOffset` (page 1696)
- `stateImageWidth` (page 1703)
- `imageAndTitleWidth` (page 1691)

Declared In
`NSMenuView.h`

locationForSubmenu:

Returns the origin of the submenu view's window.

```
- (NSPoint)locationForSubmenu:(NSMenu *)aSubmenu
```

Discussion
The *aSubmenu* parameter specifies the submenu being positioned and must belong to a menu item of this menu view. This method positions the submenu adjacent to its menu item as well as possible given the type of menu and the space constraints of the user's screen.

If any changes have been made to the menu's contents, this method invokes `sizeToFit` (page 1703) to update the menu view information.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
`- setWindowFrameForAttachingToRect:onScreen:preferredEdge:popUpSelectedItem:` (page 1702)
`- sizeToFit` (page 1703)

Declared In
`NSMenuView.h`

menu

Returns the menu object associated with this menu view.

```
- (NSMenu *)menu
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
`- setMenu:` (page 1700)

Declared In
`NSMenuView.h`

menuItemCellForItemAtIndex:

Returns the menu item cell at the specified location.

```
- (NSMenuItemCell *)menuItemCellForItemAtIndex:(NSInteger)index
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
`- setMenuItemCell:forItemAtIndex:` (page 1701)
`- sizeToFit` (page 1703)

Declared In
`NSMenuView.h`

needsSizing

Returns YES if the menu view needs to be resized due to changes in the menu object, NO otherwise.

```
- (BOOL)needsSizing
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- setNeedsSizing: (page 1702)

Declared In
NSMenuView.h

performActionWithHighlightingForItemAtIndex:

Uses the associated menu object to perform the action associated with the specified item when a key equivalent is pressed.

```
- (void)performActionWithHighlightingForItemAtIndex:(NSInteger)index
```

Discussion
Because the menu item at index might not currently be visible, this method provides visual feedback by highlighting the nearest visible parent menu item before performing the action. After the action has been sent, this method removes the highlighting for the menu item.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- performActionForItemAtIndex: (page 1624) (NSMenu)

Declared In
NSMenuView.h

rectOfItemAtIndex:

Returns the drawing rectangle of the specified menu item.

```
- (NSRect)rectOfItemAtIndex:(NSInteger)index
```

Discussion
The drawing rectangle may not be the same width or height as the actual menu and in fact is typically smaller to account for borders drawn by the menu view.

If any changes have been made to the menu's contents, this method invokes sizeToFit (page 1703) to update the menu view information.

Availability
Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

See Also
- `innerRect` (page 1693)
- `needsSizing` (page 1698)
- `sizeToFit` (page 1703)

Declared In
`NSMenuView.h`

setFont:

Sets the default font to use when drawing the menu text.

```
- (void)setFont:(NSFont *)font
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `font` (page 1690)

Declared In
`NSMenuView.h`

setHighlightedItemIndex:

Highlights the menu item at a specific location.

```
- (void)setHighlightedItemIndex:(NSInteger)index
```

Discussion
Specify –1 for *index* to remove all highlighting from the menu.

The rectangle of the menu item is marked as invalid and is redrawn the next time the event loop comes around. If another menu item was previously highlighted, that menu item is redrawn without highlights when the event loop comes around again.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `setNeedsDisplayForItemAtIndex:` (page 1701)
- `highlightedItemIndex` (page 1690)

Declared In
`NSMenuView.h`

setHorizontal:

Sets the orientation of the menu.

```
- (void)setHorizontal:(BOOL)flag
```

Discussion
If *flag* is YES, the menu's items are displayed horizontally; otherwise the menu's items are displayed vertically.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- isHorizontal (page 1694)

Declared In
NSMenuView.h

setHorizontalEdgePadding:

Sets the horizontal padding for menu item components.

```
- (void)setHorizontalEdgePadding:(CGFloat)pad
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- horizontalEdgePadding (page 1691)

Declared In
NSMenuView.h

setMenu:

Sets the menu to be displayed in the receiver

```
- (void)setMenu:(NSMenu *)menu
```

Discussion
This method invokes the setNeedsSizing: (page 1702) method to force the menu view's layout to be recalculated before drawing.

This method adds the menu view to the new NSMenu object's list of observers. The notifications this method establishes notify this menu view when menu items in the NSMenu object are added, removed, or changed. This method removes the menu view from its previous NSMenu object's list of observers.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `setNeedsSizing:` (page 1702)
- `itemAdded:` (page 1694)
- `itemRemoved:` (page 1695)
- `itemChanged:` (page 1695)

Declared In
`NSMenuView.h`

setMenuItemCell:forItemAtIndex:

Replaces the menu item cell at a specific location.

```
- (void)setMenuItemCell:(NSMenuItemCell *)cell forItemAtIndex:(NSInteger)index
```

Discussion
This method does not change the contents of the menu itself; it changes only the cell used to display the menu item at *index*. The old cell is released, and both the new cell and the menu view are marked as needing resizing.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `menuItemCellForItemAtIndex:` (page 1697)
- `setNeedsSizing:` (page 1702)

Declared In
`NSMenuView.h`

setNeedsDisplayForItemAtIndex:

Adds the region occupied by the menu item at a specific location to the menu view's invalid region.

```
- (void)setNeedsDisplayForItemAtIndex:(NSInteger)index
```

Discussion
The region to be redrawn includes the space occupied by the menu borders. This invalid region is redrawn the next time the event loop comes around.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `rectOfItemAtIndex:` (page 1698)
- `setNeedsDisplayInRect:` (page 3225) (NSView)

Declared In
`NSMenuView.h`

setNeedsSizing:

Sets a flag that indicates whether the layout is invalid and needs resizing.

```
- (void)setNeedsSizing:(BOOL)flag
```

Discussion
If *flag* is YES, the menu contents have changed or the menu appearance has changed. This method is used internally; you should not need to invoke it directly unless you are implementing a subclass that can cause the layout to become invalid.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- sizeToFit (page 1703)

Declared In
NSMenuView.h

setWindowFrameForAttachingToRect:onScreen:preferredEdge:popUpSelectedItem:

Causes the menu view to resize its window so its frame is the appropriate size to attach to a specified rectangle within the screen.

```
- (void)setWindowFrameForAttachingToRect:(NSRect)screenRect onScreen:(NSScreen
    *)screen preferredEdge:(NSRectEdge)edge
    popUpSelectedItem:(NSInteger)selectedItemIndex
```

Discussion
If *selectedItemIndex* contains a value other than –1, this method attempts to position the menu such that the item at *selectedItemIndex* appears on top of *screenRect*.

The *selectedItemIndex* parameter specifies the amount by which the selected item's rectangle overlaps *screenRect*.

If the preferred edge, *edge*, cannot be honored, because there is not enough room, the opposite edge is used. If the rectangle does not completely fit either edge, this method uses the edge where there is more room.

If any changes have been made to the menu's contents, this method invokes sizeToFit (page 1703) to update the menu view information.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- sizeToFit (page 1703)

Declared In
NSMenuView.h

sizeToFit

Used internally by the menu view to cache information about the menu item geometry.

```
- (void)sizeToFit
```

Discussion
This cache is updated as necessary when menu items are added, removed, or changed.

The geometry of each menu item is determined by asking its corresponding menu item cell. The menu item cell is obtained from the `menuItemCellForItemAtIndex:` (page 1697) method.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `setNeedsSizing:` (page 1702)
- `menuItemCellForItemAtIndex:` (page 1697)

Declared In
`NSMenuView.h`

stateImageOffset

Returns the offset to the space reserved for state images of this menu.

```
- (CGFloat)stateImageOffset
```

Discussion
The offset is used for all menu items of the menu.

If any changes have been made to the menu's contents, this method invokes `sizeToFit` (page 1703) to update the menu view information.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
- `horizontalEdgePadding` (page 1691)
- `setHorizontalEdgePadding:` (page 1700)
- `sizeToFit` (page 1703)

Declared In
`NSMenuView.h`

stateImageWidth

Returns the maximum width of the state images used by this menu.

```
- (CGFloat)stateImageWidth
```

Discussion
The width is used for all menu items of the menu.

If any changes have been made to the menu's contents, this method invokes `sizeToFit` (page 1703) to update the menu view information.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
`- sizeToFit` (page 1703)

Declared In
`NSMenuView.h`

trackWithEvent:

Handles events sent to this menu view.

```
- (BOOL)trackWithEvent:(NSEvent *)event
```

Discussion
If *event* is a mouse event, this method tracks the cursor position in the menu and displays the menus as appropriate. This method also handles mouse clicks that result in the selection of a menu item, in which case the menu item's action is performed.

You should not need to use this method directly.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`NSMenuView.h`

update

Asks the associated menu object to update itself.

```
- (void)update
```

Discussion
If any changes have been made to the menu's contents, this method invokes `sizeToFit` (page 1703) to update the menu view's layout.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

See Also
`- sizeToFit` (page 1703)
`- setNeedsSizing:` (page 1702)

- update (page 1635) (NSMenu)

Declared In
NSMenuView.h

NSMutableAttributedString Additions Reference

Inherits from	NSAttributedString : NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSAttributedString.h AppKit/NSTextAttachment.h
Companion guide	Attributed String Programming Guide

Overview

Additions to the NSMutableAttributedString class primarily involve setting graphical attributes, such as font, super- or subscripting, and alignment, and making these attributes consistent after changes.

Tasks

Changing Attributes

- `applyFontTraits:range:` (page 1708)
  Applies the font attributes specified by *mask* to the characters in *aRange*.
- `setAlignment:range:` (page 1712)
  Sets the alignment characteristic of the paragraph style attribute for the characters in *aRange* to *alignment*.
- `setBaseWritingDirection:range:` (page 1712)
  Sets the base writing direction for the characters in *range* to *writingDirection*.
- `subscriptRange:` (page 1713)
  Decrements the value of the superscript attribute for characters in *aRange* by 1.
- `superscriptRange:` (page 1713)
  Increments the value of the superscript attribute for characters in *aRange* by 1.
- `unscriptRange:` (page 1714)
  Removes the superscript attribute from the characters in *aRange*.

Updating Attachment Contents

- `updateAttachmentsFromPath:` (page 1714)
  Updates all attachments based on files contained in the RTFD file package at *path*.

Fixing Attributes After Changes

- `fixAttributesInRange:` (page 1709)
  Invokes the other `fix...` methods, allowing you to clean up an attributed string with a single message.
- `fixAttachmentAttributeInRange:` (page 1709)
  Cleans up attachment attributes in *aRange*, removing all attachment attributes assigned to characters other than `NSAttachmentCharacter`.
- `fixFontAttributeInRange:` (page 1709)
  Fixes the font attribute in *aRange*, assigning default fonts to characters with illegal fonts for their scripts and otherwise correcting font attribute assignments.
- `fixParagraphStyleAttributeInRange:` (page 1710)
  Fixes the paragraph style attributes in *aRange*, assigning the first paragraph style attribute value in each paragraph to all characters of the paragraph.

Reading Content

- `readFromData:options:documentAttributes:` (page 1710)
  Sets the contents of the receiver from the stream at *data*.
- `readFromData:options:documentAttributes:error:` (page 1711)
  Sets the contents of the receiver from the stream at *data*.
- `readFromURL:options:documentAttributes:` (page 1711)
  Sets the contents of receiver from the file at *url*.
- `readFromURL:options:documentAttributes:error:` (page 1712)
  Sets the contents of receiver from the file at *url*.

Instance Methods

applyFontTraits:range:

Applies the font attributes specified by *mask* to the characters in *aRange*.

```
- (void)applyFontTraits:(NSFontTraitMask)mask range:(NSRange)aRange
```

Discussion
See the NSFontManager class specification for a description of the font traits available. Raises an `NSRangeException` if any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlignment:range:` (page 1712)

Declared In
`NSAttributedString.h`

fixAttachmentAttributeInRange:

Cleans up attachment attributes in *aRange*, removing all attachment attributes assigned to characters other than `NSAttachmentCharacter`.

```
- (void)fixAttachmentAttributeInRange:(NSRange)aRange
```

Discussion
Raises an `NSRangeException` if any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `fixFontAttributeInRange:` (page 1709)
- `fixParagraphStyleAttributeInRange:` (page 1710)
- `fixAttributesInRange:` (page 1709)

Declared In
`NSAttributedString.h`

fixAttributesInRange:

Invokes the other `fix...` methods, allowing you to clean up an attributed string with a single message.

```
- (void)fixAttributesInRange:(NSRange)aRange
```

Discussion
Raises an `NSRangeException` if any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `fixAttachmentAttributeInRange:` (page 1709)
- `fixFontAttributeInRange:` (page 1709)
- `fixParagraphStyleAttributeInRange:` (page 1710)

Declared In
`NSAttributedString.h`

fixFontAttributeInRange:

Fixes the font attribute in *aRange*, assigning default fonts to characters with illegal fonts for their scripts and otherwise correcting font attribute assignments.

```
- (void)fixFontAttributeInRange:(NSRange)aRange
```

Discussion
For example, Kanji characters assigned a Latin font are reassigned an appropriate Kanji font. Raises an `NSRangeException` if any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `fixParagraphStyleAttributeInRange:` (page 1710)
- `fixAttachmentAttributeInRange:` (page 1709)
- `fixAttributesInRange:` (page 1709)

Declared In
`NSAttributedString.h`

fixParagraphStyleAttributeInRange:

Fixes the paragraph style attributes in *aRange*, assigning the first paragraph style attribute value in each paragraph to all characters of the paragraph.

```
- (void)fixParagraphStyleAttributeInRange:(NSRange)aRange
```

Discussion
This method extends the range as needed to cover the last paragraph partially contained. A paragraph is delimited by any of these characters, the longest possible sequence being preferred to any shorter:

U+000D (`\r` or CR)
U+000A (`\n` or LF)
U+2028 (Unicode line separator)
U+2029 (Unicode paragraph separator) `\r\n`, in that order (also known as CRLF)

Raises an `NSRangeException` if any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `fixFontAttributeInRange:` (page 1709)
- `fixAttachmentAttributeInRange:` (page 1709)
- `fixAttributesInRange:` (page 1709)

Declared In
`NSAttributedString.h`

readFromData:options:documentAttributes:

Sets the contents of the receiver from the stream at *data*.

```
- (BOOL)readFromData:(NSData *)data options:(NSDictionary *)options
    documentAttributes:(NSDictionary **)dict
```

Discussion
options can contain one of the values described in the "Constants" (page 271) section of *NSAttributedString Application Kit Additions Reference* ("Option keys for importing documents").

On return, the *documentAttributes* dictionary (if provided) contains the various keys described in the "Constants" (page 271) section of *NSAttributedString Application Kit Additions Reference*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSAttributedString.h

readFromData:options:documentAttributes:error:

Sets the contents of the receiver from the stream at *data*.

```
- (BOOL)readFromData:(NSData *)data options:(NSDictionary *)opts
    documentAttributes:(NSDictionary **)dict error:(NSError **)error
```

Discussion
opts can contain one of the values described in the "Constants" (page 271) section of *NSAttributedString Application Kit Additions Reference* ("Option keys for importing documents").

On return, the *documentAttributes* dictionary (if provided) contains the various keys described in the "Constants" (page 271) section of *NSAttributedString Application Kit Additions Reference*. If unsuccessful, returns NO , after setting *error* to point to an NSError object that encapsulates the reason why the attributed string object could not be created.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSAttributedString.h

readFromURL:options:documentAttributes:

Sets the contents of receiver from the file at *url*.

```
- (BOOL)readFromURL:(NSURL *)url options:(NSDictionary *)options
    documentAttributes:(NSDictionary **)documentAttributes
```

Discussion
Filter services can be used to convert the contents of the URL into a format recognized by Cocoa. *options* can contain one of the values described in the "Constants" (page 271) section of *NSAttributedString Application Kit Additions Reference* ("Option keys for importing documents").

On return, the *documentAttributes* dictionary (if provided) contains the various keys described in the "Constants" (page 271) section of *NSAttributedString Application Kit Additions Reference*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSAttributedString.h

readFromURL:options:documentAttributes:error:

Sets the contents of receiver from the file at *url*.

```
- (BOOL)readFromURL:(NSURL *)url options:(NSDictionary *)opts
    documentAttributes:(NSDictionary **)dict error:(NSError **)error
```

Discussion
Filter services can be used to convert the contents of the URL into a format recognized by Cocoa. *opts* can contain one of the values described in the "Constants" (page 271) section of *NSAttributedString Application Kit Additions Reference* ("Option keys for importing documents").

On return, the *dict* dictionary (if provided) contains the various keys described in the "Constants" (page 271) section of *NSAttributedString Application Kit Additions Reference*. If unsuccessful, returns `NO`, after setting *error* to point to an `NSError` object that encapsulates the reason why the attributed string object could not be created.

For RTF formatted files, the contents of the file are appended to the previous string instead of replacing the previous string. Therefore, when using this method with existing content it's best to clear the content away explicitly.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSAttributedString.h`

setAlignment:range:

Sets the alignment characteristic of the paragraph style attribute for the characters in *aRange* to *alignment*.

```
- (void)setAlignment:(NSTextAlignment)alignment range:(NSRange)aRange
```

Discussion
When attribute fixing takes place, this change will affect only paragraphs whose first character was included in *aRange*. Raises an `NSRangeException` if any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `applyFontTraits:range:` (page 1708)
- `fixParagraphStyleAttributeInRange:` (page 1710)

Declared In
`NSAttributedString.h`

setBaseWritingDirection:range:

Sets the base writing direction for the characters in *range* to *writingDirection*.

```
- (void)setBaseWritingDirection:(NSWritingDirection)writingDirection
    range:(NSRange)range
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSAttributedString.h

subscriptRange:

Decrements the value of the superscript attribute for characters in *aRange* by 1.

```
- (void)subscriptRange:(NSRange)aRange
```

Discussion
Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- superscriptRange: (page 1713)
- unscriptRange: (page 1714)

Related Sample Code
CoreRecipes

Declared In
NSAttributedString.h

superscriptRange:

Increments the value of the superscript attribute for characters in *aRange* by 1.

```
- (void)superscriptRange:(NSRange)aRange
```

Discussion
Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- subscriptRange: (page 1713)
- unscriptRange: (page 1714)

Related Sample Code
CoreRecipes

Declared In
NSAttributedString.h

unscriptRange:

Removes the superscript attribute from the characters in *aRange*.

```
- (void)unscriptRange:(NSRange)aRange
```

Discussion
Raises an `NSRangeException` if any part of *aRange* lies beyond the end of the receiver's characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `subscriptRange:` (page 1713)
- `superscriptRange:` (page 1713)

Declared In
`NSAttributedString.h`

updateAttachmentsFromPath:

Updates all attachments based on files contained in the RTFD file package at *path*.

```
- (void)updateAttachmentsFromPath:(NSString *)path
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `updateFromPath:` (page 1137) (`NSFileWrapper`)

Declared In
`NSTextAttachment.h`

NSMutableParagraphStyle Class Reference

Inherits from	NSParagraphStyle : NSObject
Conforms to	NSCoding (NSParagraphStyle) NSCopying (NSParagraphStyle) NSMutableCopying (NSParagraphStyle) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSParagraphStyle.h
Companion guide	Rulers and Paragraph Styles
Related sample code	FilterDemo IBFragmentView ImageBrowserViewAppearance ImageKitDemo Quartz Composer WWDC 2005 TextEdit

Overview

NSMutableParagraphStyle adds methods to its superclass, NSParagraphStyle, for changing the values of the subattributes in a paragraph style attribute. See the NSParagraphStyle and NSAttributedString specifications for more information.

Important: A paragraph style object should not be mutated after adding it to an attributed string; doing so can cause your program to crash.

Tasks

Setting Tab Stops

`- setTabStops:` (page 1724)
Replaces the tab stops in the receiver with *tabStops*.

- `addTabStop:` (page 1717)
  Adds *tabStop* to the receiver.
- `removeTabStop:` (page 1717)
  Removes the first text tab whose location and type are equal to those of *tabStop*.

Setting Other Style Information

- `setParagraphStyle:` (page 1724)
  Replaces the subattributes of the receiver with those in *aStyle*.
- `setAlignment:` (page 1718)
  Sets the alignment of the receiver to *alignment*.
- `setFirstLineHeadIndent:` (page 1719)
  Sets the distance in points from the leading margin of a text container to the beginning of the paragraph's first line to *aFloat*.
- `setHeadIndent:` (page 1720)
  Sets the distance in points from the leading margin of a text container to the beginning of lines other than the first to *aFloat*.
- `setTailIndent:` (page 1725)
  Sets the distance in points from the margin of a text container to the end of lines to *aFloat*.
- `setLineBreakMode:` (page 1721)
  Sets the mode used to break lines in a layout container to *mode*.
- `setMaximumLineHeight:` (page 1722)
  Sets the maximum height that any line in the paragraph style will occupy, regardless of the font size or size of any attached graphic, to *aFloat*.
- `setMinimumLineHeight:` (page 1723)
  Sets the minimum height that any line in the paragraph style will occupy, regardless of the font size or size of any attached graphic, to *aFloat*.
- `setLineSpacing:` (page 1722)
  Sets the space in points added between lines within the paragraph to *aFloat*.
- `setParagraphSpacing:` (page 1723)
  Sets the space added at the end of the paragraph to separate it from the following paragraph to *aFloat*.
- `setBaseWritingDirection:` (page 1718)
  Sets the base writing direction for the receiver.
- `setLineHeightMultiple:` (page 1722)
  Sets the line height multiple for the receiver.
- `setParagraphSpacingBefore:` (page 1724)
  Sets the distance between the paragraph's top and the beginning of its text content
- `setDefaultTabInterval:` (page 1719)
  Sets the default tab interval for the receiver.

Setting Text Blocks and Lists

- `setTextBlocks:` (page 1725)
  Sets the text blocks containing the paragraph, nested from outermost to innermost to *array*.
- `setTextLists:` (page 1726)

Controlling Hyphenation and Truncation

- `setHyphenationFactor:` (page 1720)
  Specifies the threshold for hyphenation.
- `setTighteningFactorForTruncation:` (page 1726)
  Specifies the threshold for using tightening as an alternative to truncation.

Setting HTML Header Level

- `setHeaderLevel:` (page 1720)
  Specifies whether the paragraph is to be treated as a header for purposes of HTML generation.

Instance Methods

addTabStop:

Adds *tabStop* to the receiver.

```
- (void)addTabStop:(NSTextTab *)tabStop
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `removeTabStop:` (page 1717)
- `setTabStops:` (page 1724)
- `tabStops` (page 1876) (NSParagraphStyle)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSParagraphStyle.h`

removeTabStop:

Removes the first text tab whose location and type are equal to those of *tabStop*.

```
- (void)removeTabStop:(NSTextTab *)tabStop
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addTabStop:` (page 1717)
- `setTabStops:` (page 1724)
- `tabStops` (page 1876) (NSParagraphStyle)

Declared In
`NSParagraphStyle.h`

setAlignment:

Sets the alignment of the receiver to *`alignment`*.

```
- (void)setAlignment:(NSTextAlignment)alignment
```

Discussion
`alignment` may be one of:

```
NSLeftTextAlignment
NSRightTextAlignment
NSCenterTextAlignment
NSJustifiedTextAlignment
NSNaturalTextAlignment
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alignment` (page 1870) (NSParagraphStyle)

Related Sample Code
FilterDemo
IBFragmentView
ImageBrowserViewAppearance
ImageKitDemo
iSpend

Declared In
`NSParagraphStyle.h`

setBaseWritingDirection:

Sets the base writing direction for the receiver.

```
- (void)setBaseWritingDirection:(NSWritingDirection)writingDirection
```

Discussion
It can be `NSWritingDirectionNaturalDirection`, `NSWritingDirectionLeftToRight`, or `NSWritingDirectionRightToLeft`. If you specify `NSWritingDirectionNaturalDirection`, the receiver resolves the writing direction to either `NSWritingDirectionLeftToRight` or `NSWritingDirectionRightToLeft`, depending on the direction for the user's language preference setting.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ defaultWritingDirectionForLanguage:` (page 1870) (NSParagraphStyle)
`- baseWritingDirection` (page 1871) (NSParagraphStyle)

Related Sample Code
PhotoSearch

Declared In
`NSParagraphStyle.h`

setDefaultTabInterval:

Sets the default tab interval for the receiver.

```
- (void)setDefaultTabInterval:(CGFloat)aFloat
```

Discussion
Tabs after the last specified in `tabStops` (page 1876) are placed at integral multiples of this distance. This value must be nonnegative.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- defaultTabInterval` (page 1871) (NSParagraphStyle)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSParagraphStyle.h`

setFirstLineHeadIndent:

Sets the distance in points from the leading margin of a text container to the beginning of the paragraph's first line to *aFloat*.

```
- (void)setFirstLineHeadIndent:(CGFloat)aFloat
```

Discussion
This value must be nonnegative.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHeadIndent:` (page 1720)
- `setTailIndent:` (page 1725)
- `firstLineHeadIndent` (page 1872) (NSParagraphStyle)

Declared In
`NSParagraphStyle.h`

setHeaderLevel:

Specifies whether the paragraph is to be treated as a header for purposes of HTML generation.

```
- (void)setHeaderLevel:(NSInteger)level
```

Discussion
Should be set to 0 (the default value) if the paragraph is not a header, or from 1 through 6 if the paragraph is to be treated as a header.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `headerLevel` (page 1872) (NSParagraphStyle)

Declared In
`NSParagraphStyle.h`

setHeadIndent:

Sets the distance in points from the leading margin of a text container to the beginning of lines other than the first to *aFloat*.

```
- (void)setHeadIndent:(CGFloat)aFloat
```

Discussion
This value must be nonnegative.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFirstLineHeadIndent:` (page 1719)
- `setTailIndent:` (page 1725)
- `headIndent` (page 1872) (NSParagraphStyle)

Declared In
`NSParagraphStyle.h`

setHyphenationFactor:

Specifies the threshold for hyphenation.

```
- (void)setHyphenationFactor:(float)aFactor
```

Discussion
Valid values lie between 0.0 and 1.0 inclusive. The default value is 0.0. Hyphenation is attempted when the ratio of the text width (as broken without hyphenation) to the width of the line fragment is less than the hyphenation factor. When the paragraph's hyphenation factor is 0.0, the layout manager's hyphenation factor is used instead. When both are 0.0, hyphenation is disabled.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `hyphenationFactor` (page 1873) (NSParagraphStyle)

Declared In
`NSParagraphStyle.h`

setLineBreakMode:

Sets the mode used to break lines in a layout container to *mode*.

```
- (void)setLineBreakMode:(NSLineBreakMode)mode
```

Discussion
The *mode* parameter may be one of:

```
NSLineBreakByWordWrapping
NSLineBreakByCharWrapping
NSLineBreakByClipping
NSLineBreakByTruncatingHead
NSLineBreakByTruncatingTail
NSLineBreakByTruncatingMiddle
```

See the description of `lineBreakMode` (page 1873) in the NSParagraphStyle class specification for descriptions of these values.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Cocoa Tips and Tricks
IBFragmentView
ImageBrowserViewAppearance
ImageKitDemo
PDFKitLinker2

Declared In
`NSParagraphStyle.h`

setLineHeightMultiple:

Sets the line height multiple for the receiver.

```
- (void)setLineHeightMultiple:(CGFloat)aFloat
```

Discussion
The natural line height of the receiver is multiplied by this factor before being constrained by minimum and maximum line height. This value must be nonnegative.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `lineHeightMultiple` (page 1873) (NSParagraphStyle)

Declared In
`NSParagraphStyle.h`

setLineSpacing:

Sets the space in points added between lines within the paragraph to *aFloat*.

```
- (void)setLineSpacing:(CGFloat)aFloat
```

Discussion
This value must be nonnegative.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMaximumLineHeight:` (page 1722)
- `setMinimumLineHeight:` (page 1723)
- `setParagraphSpacing:` (page 1723)
- `lineSpacing` (page 1874) (NSParagraphStyle)

Related Sample Code
TipWrapper

Declared In
`NSParagraphStyle.h`

setMaximumLineHeight:

Sets the maximum height that any line in the paragraph style will occupy, regardless of the font size or size of any attached graphic, to *aFloat*.

```
- (void)setMaximumLineHeight:(CGFloat)aFloat
```

Discussion
Glyphs and graphics exceeding this height will overlap neighboring lines; however, a maximum height of 0 implies no line height limit. This value must be nonnegative.

Although this limit applies to the line itself, line spacing adds extra space between adjacent lines.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMinimumLineHeight:` (page 1723)
- `setLineSpacing:` (page 1722)
- `maximumLineHeight` (page 1874) (NSParagraphStyle)

Declared In
`NSParagraphStyle.h`

setMinimumLineHeight:

Sets the minimum height that any line in the paragraph style will occupy, regardless of the font size or size of any attached graphic, to *aFloat*.

```
- (void)setMinimumLineHeight:(CGFloat)aFloat
```

Discussion
This value must be nonnegative.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMaximumLineHeight:` (page 1722)
- `setLineSpacing:` (page 1722)
- `minimumLineHeight` (page 1875) (NSParagraphStyle)

Declared In
`NSParagraphStyle.h`

setParagraphSpacing:

Sets the space added at the end of the paragraph to separate it from the following paragraph to *aFloat*.

```
- (void)setParagraphSpacing:(CGFloat)aFloat
```

Discussion
This value must be nonnegative.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setLineSpacing:` (page 1722)
- `setParagraphSpacingBefore:` (page 1724)
- `paragraphSpacing` (page 1875) (NSParagraphStyle)

Declared In
NSParagraphStyle.h

setParagraphSpacingBefore:

Sets the distance between the paragraph's top and the beginning of its text content

```
- (void)setParagraphSpacingBefore:(CGFloat)aFloat
```

Discussion
. This value must be nonnegative.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setParagraphSpacing: (page 1723)
- paragraphSpacingBefore (page 1876) (NSParagraphStyle)

Declared In
NSParagraphStyle.h

setParagraphStyle:

Replaces the subattributes of the receiver with those in *aStyle*.

```
- (void)setParagraphStyle:(NSParagraphStyle *)aStyle
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
FilterDemo

Declared In
NSParagraphStyle.h

setTabStops:

Replaces the tab stops in the receiver with *tabStops*.

```
- (void)setTabStops:(NSArray *)tabStops
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- addTabStop: (page 1717)
- removeTabStop: (page 1717)
- tabStops (page 1876) (NSParagraphStyle)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSParagraphStyle.h`

setTailIndent:

Sets the distance in points from the margin of a text container to the end of lines to *aFloat*.

```
- (void)setTailIndent:(CGFloat)aFloat
```

Discussion
If positive, this is the distance from the leading margin (for example, the left margin in left-to-right text). That is, it's the absolute line width. If 0 or negative, it's the distance from the trailing margin—the value is added to the line width.

For example, to create a paragraph style that fits exactly in a 2-inch wide container, set its head indent to 0.0 and its tail indent to 0.0. To create a paragraph style with quarter-inch margins, set its head indent to 0.25 and its tail indent to –0.25.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHeadIndent:` (page 1720)
- `setFirstLineHeadIndent:` (page 1719)
- `tailIndent` (page 1877) (NSParagraphStyle)

Declared In
`NSParagraphStyle.h`

setTextBlocks:

Sets the text blocks containing the paragraph, nested from outermost to innermost to *array*.

```
- (void)setTextBlocks:(NSArray *)array
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `textBlocks` (page 1877) (NSParagraphStyle)

Related Sample Code
iSpend

Declared In
`NSParagraphStyle.h`

setTextLists:

```
- (void)setTextLists:(NSArray *)array
```

Discussion
Sets the text lists containing the paragraph, nested from outermost to innermost, to *array*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- textLists (page 1877) (NSParagraphStyle)

Declared In
NSParagraphStyle.h

setTighteningFactorForTruncation:

Specifies the threshold for using tightening as an alternative to truncation.

```
- (void)setTighteningFactorForTruncation:(float)aFactor
```

Discussion
When the line break mode specifies truncation, the text system attempts to tighten intercharacter spacing as an alternative to truncation, provided that the ratio of the text width to the line fragment width does not exceed 1.0 + the value returned by tighteningFactorForTruncation (page 1878). Otherwise the text is truncated at a location determined by the line break mode. The default value is 0.05. This method accepts positive and negative values. Values less than or equal to 0.0 result in not tightening.

Availability
Available in Mac OS X v10.4 and later.

See Also
- tighteningFactorForTruncation (page 1878) (NSParagraphStyle)

Declared In
NSParagraphStyle.h

NSNib Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSNib.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Resource Programming Guide
Related sample code	Aperture Edit Plugin - Borders & Titles Aperture Image Resizer Cocoa Tips and Tricks CoreRecipes Departments and Employees

Overview

Instances of the `NSNib` class serve as object wrappers, or containers, for Interface Builder nib files. An `NSNib` object keeps the contents of a nib file resident in memory, ready for unarchiving and instantiation.

When you create an `NSNib` object using the contents of a nib file, the object loads the contents of the referenced nib bundle—the object graph as well as any images and sounds—into memory; but it does not yet unarchive it. To unarchive all of the nib data and thus truly instantiate the nib you must call one of the `instantiate...` methods of `NSNib`.

During the instantiation process, each object in the archive is unarchived and then initialized using the method befitting its type. View classes are initialized using their `initWithFrame:` method. Custom objects are initialized using their `init` method. In the case of Cocoa views (and custom views that have options on an associated Interface Builder palette) the initialization process also reads in any values set by the user in Interface Builder.

Once all objects have been instantiated and initialized from the archive, the nib loading code attempts to reestablish the connections between each object's outlets and the corresponding target objects. If your custom objects have outlets, the `NSNib` object attempts to reestablish any connections you created in Interface Builder. It starts by trying to establish the connections using your object's own methods first. For each outlet that needs a connection, the `NSNib` object looks for a method of the form `set`*OutletName*`:` in your object. If that method exists, the nib object calls it, passing the target object as a parameter. If you did

not define a setter method with that exact name, the `NSNib` object searches the object for an instance variable (of type `IBOutlet id`) with the corresponding outlet name and tries to set its value directly. If an instance variable with the correct name cannot be found, initialization of that connection does not occur.

After all objects have been initialized and their connections reestablished, each object receives an `awakeFromNib` message. You can override this method in your custom objects to perform any additional initialization.

Subclassing Notes

You can subclass `NSNib` if you want to extend or specialize nib-loading behavior. For example, you could create a custom `NSNib` subclass that performs some post-processing on the top-level objects returned from the `instantiateNib...` methods. If you want to modify how nib instantiations are performed, it is recommended that you override the primitive method `instantiateNibWithExternalNameTable:` (page 1730). Note that the instance variables of `NSNib` are private and thus are not available to subclasses. Any override of `initWithContentsOfURL:` (page 1729) or `initWithNibNamed:bundle:` (page 1729) should first invoke the superclass implementation.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Initializing a Nib

`- initWithContentsOfURL:` (page 1729)
Returns an `NSNib` object initialized to the nib file at the specified URL.

`- initWithNibNamed:bundle:` (page 1729)
Returns an `NSNib` object initialized to the nib file in the specified bundle.

Instantiating a Nib

`- instantiateNibWithOwner:topLevelObjects:` (page 1730)
Unarchives and instantiates the in-memory contents of the receiver's nib file, creating a distinct object tree and set of top level objects.

`- instantiateNibWithExternalNameTable:` (page 1730)
Unarchives and instantiates the in-memory contents of the receiver's nib file, creating a distinct object tree and top level objects.

Instance Methods

initWithContentsOfURL:

Returns an `NSNib` object initialized to the nib file at the specified URL.

```
- (id)initWithContentsOfURL:(NSURL *)nibFileURL
```

Parameters

nibFileURL

The location of the nib file.

Return Value

The initialized `NSNib` object or `nil` if there were errors during initialization or the nib file could not be located.

Discussion

When you instantiate the nib objects later, the `NSNib` object looks for an appropriate bundle from which to search for any additional resources referenced by the nib. Because you do not specify a bundle directory when calling this method, the receiver uses the bundle associated with the class of the nib file's owner. If the nib file does not have an owner, the receiver uses the application's main bundle instead.

Availability

Available in Mac OS X v10.3 and later.

See Also

`NSURL` class (Foundation)

Declared In

`NSNib.h`

initWithNibNamed:bundle:

Returns an `NSNib` object initialized to the nib file in the specified bundle.

```
- (id)initWithNibNamed:(NSString *)nibName bundle:(NSBundle *)bundle
```

Parameters

nibName

The name of the nib file, without any leading path information. Inclusion of the `.nib` extension on the nib file name is optional.

bundle

The bundle in which to search for the nib file. If you specify `nil`, this method looks for the nib file in the main bundle.

Return Value

The initialized `NSNib` object or `nil` if there were errors during initialization or the nib file could not be located.

Discussion

The `NSNib` object looks for the nib file in the bundle's language-specific project directories first, followed by the `Resources` directory.

After the nib file has been loaded, the `NSNib` object uses the bundle's resource map to locate additional resources referenced by the nib. If you specified `nil` for the *`bundle`* parameter, the `NSNib` object looks for those resources in the bundle associated with the class of the nib file's owner instead. If the nib file does not have an owner, the `NSNib` object looks for additional resources in the application's main bundle.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
Aperture Image Resizer
Cocoa Tips and Tricks
CoreRecipes
Departments and Employees

Declared In
`NSNib.h`

instantiateNibWithExternalNameTable:

Unarchives and instantiates the in-memory contents of the receiver's nib file, creating a distinct object tree and top level objects.

```
- (BOOL)instantiateNibWithExternalNameTable:(NSDictionary *)externalNameTable
```

Parameters

`externalNameTable`
A dictionary containing entries for the nib file's owner and top-level objects. See the discussion for more information.

Return Value
`YES` if the nib file's contents were instantiated successfully; otherwise, `NO`.

Discussion
This is the primitive method for performing instantiations of a nib file. You may use this method to instantiate a nib file multiple times. Each instantiation of the nib must have a distinct owner object that is responsible for the resulting object tree.

If the nib file requires an owner, the *`externalNameTable`* parameter must contain the object representing the nib file's owner (associated with the `NSNibOwner` key). The parameter may optionally include an `NSMutableArray` object to be populated with the top-level objects nib file (associated with the `NSNibTopLevelObjects` key).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSNib.h`

instantiateNibWithOwner:topLevelObjects:

Unarchives and instantiates the in-memory contents of the receiver's nib file, creating a distinct object tree and set of top level objects.

```
- (BOOL)instantiateNibWithOwner:(id)owner topLevelObjects:(NSArray **)topLevelObjects
```

Parameters

owner

The object to use as the owner of the nib file. If the nib file has an owner, you must specify a valid object for this parameter.

topLevelObjects

On input, a variable capable of holding an `NSArray` object. On output, this variable contains an autoreleased `NSArray` object containing the top-level objects from the nib file. You may specify `nil` for this parameter if you are not interested in the top-level objects.

Return Value

`YES` if the nib file's contents were instantiated successfully; otherwise, `NO`.

Discussion

You may use this method to instantiate a nib file multiple times. This is a convenience method that composes the name-table dictionary and invokes the `instantiateNibWithExternalNameTable:` (page 1730) method, passing it the name table.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

Cocoa Tips and Tricks

CoreRecipes

Departments and Employees

Declared In

`NSNib.h`

Constants

Nib Loading Keys

The `NSNib` class uses the following constants which are used as keys in the dictionary passed to `instantiateNibWithExternalNameTable:` (page 1730).

```
NSString *NSNibOwner;
NSString *NSNibTopLevelObjects;
```

Constants

`NSNibOwner`

The external object that is responsible for the instantiated nib.

This key is required.

Available in Mac OS X v10.3 and later.

Declared in `NSNib.h`.

`NSNibTopLevelObjects`

An `NSMutableArray` object that, if present, is populated with the top-level objects of the newly instantiated nib.

Because you must allocate this array, you are responsible for its disposal. This key is optional.

Available in Mac OS X v10.3 and later.

Declared in `NSNib.h`.

Declared In
`NSNib.h`

NSNibConnector Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSNibConnector.h
Companion guide	Resource Programming Guide

Overview

This class represents a basic connection in Interface Builder. You should not use this class directly. If you need to work with nib connections, you would use a subclass like `NSNibControlConnector` or `NSNibOutletConnector` instead. If you want to create your own type of connectors, you can also define your own custom subclasses.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Working with the Source

- `source` (page 1736)
  Returns the connector's source.
- `setSource:` (page 1736)
  Sets the connector's source to the specified object.

Working with the Destination

- `destination` (page 1734)
  Returns the connector's destination.
- `setDestination:` (page 1736)
  Sets the connector's destination to *`destination`*.

Working with the Connection

- `establishConnection` (page 1734)
  Establishes a connection between the source and destination object.
- `replaceObject:withObject:` (page 1735)
  Changes the connection's source or destination object to the specified object.
- `label` (page 1735)
  Returns the label associated with the connection.
- `setLabel:` (page 1736)
  Sets the label for the connection.

Instance Methods

destination

Returns the connector's destination.

```
- (id)destination
```

Return Value
The object that is the destination of the connection.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSNibConnector.h`

establishConnection

Establishes a connection between the source and destination object.

```
- (void)establishConnection
```

Discussion
The default implementation of this method does nothing. Subclasses must override it to establish a connection between the source and destination objects. The current label provides the description of how the two objects are connected and can be interpreted differently by different subclasses. This method is called for each connection whenever an application opens a nib file.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `label` (page 1735)
- `source` (page 1736)
- `destination` (page 1734)

Declared In
`NSNibConnector.h`

label

Returns the label associated with the connection.

```
- (NSString *)label
```

Return Value
A string containing information about the type of connection. This value can be interpreted differently by different subclasses. For example, the `NSNibControlConnector` interprets this string as the selector to call as an action method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSNibConnector.h`

replaceObject:withObject:

Changes the connection's source or destination object to the specified object.

```
- (void)replaceObject:(id)oldObject withObject:(id)newObject
```

Parameters

oldObject
The object you want to replace. This object can be either the current source object or the current destination object.

newObject
The replacement object.

Discussion
If the object in *oldObject* is not used for either the source or destination of this connection, this method does nothing.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSNibConnector.h`

setDestination:

Sets the connector's destination to *destination*.

```
- (void)setDestination:(id)destination
```

Parameters

destination

The object that is the destination of the connection.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSNibConnector.h

setLabel:

Sets the label for the connection.

```
- (void)setLabel:(NSString *)label
```

Parameters

label

A string containing information about the type of connection. This value can be interpreted differently by different subclasses. For example, the `NSNibControlConnector` interprets this string as the selector to call as an action method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSNibConnector.h

setSource:

Sets the connector's source to the specified object.

```
- (void)setSource:(id)source
```

Parameters

source

The object that is the source of the connection.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSNibConnector.h

source

Returns the connector's source.

```
- (id)source
```

Return Value
The object that is the source of the connection.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSNibConnector.h`

NSNibControlConnector Class Reference

Inherits from	NSNibConnector : NSObject
Conforms to	NSCoding (NSNibConnector) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSNibControlConnector.h
Companion guide	Resource Programming Guide

Overview

This class manages an action connection in Interface Builder. Generally, its source is a user interface item, such as a button or menu item; its destination is an object that responds to an action method; and its label is the name of the action method, with a colon at the end.

Tasks

Establishing a Connection

`- establishConnection` (page 1739)
Establishes an action connection.

Instance Methods

establishConnection

Establishes an action connection.

```
- (void)establishConnection
```

Discussion
This method establishes a connection between the source of an action and its destination. The label associated with the connection object contains the selector name to perform when the action occurs.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setSource:` (page 1736) (`NSNibConnector`)
- `setDestination:` (page 1736) (`NSNibConnector`)
- `setLabel:` (page 1736) (`NSNibConnector`)

Declared In
`NSNibControlConnector.h`

NSNibOutletConnector Class Reference

Inherits from	NSNibConnector : NSObject
Conforms to	NSCoding (NSNibConnector) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSNibOutletConnector.h
Companion guide	Resource Programming Guide

Overview

This class manages an outlet connection in Interface Builder. The label is of the form `"value"`. If the source contains a method of the form `setValue:`, then that method is called with the destination as the argument. Otherwise, if the label is the name of an instance variable in the source, then that instance variable is set to the destination.

Tasks

Establishing a Connection

`- establishConnection` (page 1741)
Establishes an outlet connection.

Instance Methods

establishConnection

Establishes an outlet connection.

```
- (void)establishConnection
```

Discussion
If the label is a method name of the form `setValue:` (where *Value* can be anything), that method is called on the source object. The parameter passed to that method is the destination object.

If the label contains the name of an outlet, this method sets the value of the outlet to the destination object. The outlet must specify an instance variable in the source object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setSource:` (page 1736) (`NSNibConnector`)
- `setDestination:` (page 1736) (`NSNibConnector`)
- `setLabel:` (page 1736) (`NSNibConnector`)

Declared In
`NSNibOutletConnector.h`

NSObjectController Class Reference

Inherits from	NSController : NSObject
Conforms to	NSCoding (NSController) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSObjectController.h
Availability	Available in Mac OS X v10.3 and later.
Companion guides	Cocoa Bindings Programming Topics Predicate Programming Guide Core Data Programming Guide
Related sample code	CIRAWFilterSample CoreRecipes DispatchFractal GridCalendar Simple Bindings Adoption

Overview

`NSObjectController` is a Cocoa bindings-compatible controller class. Properties of the content object of an instance of this class can be bound to user interface elements to access and modify their values.

By default, the content of an `NSObjectController` instance is an `NSMutableDictionary` object. This allows a single `NSObjectController` instance to be used to manage many different properties referenced by key value paths. The default content object class can be changed by calling `setObjectClass:` (page 1758), which subclassers must override.

Tasks

Initializing an Object Controller

`- initWithContent:` (page 1751)
Initializes and returns an `NSObjectController` object with the given content.

Managing Content

- `setContent:` (page 1756)
  Sets the receiver's content object.
- `content` (page 1748)
  Returns the receiver's content object.
- `setAutomaticallyPreparesContent:` (page 1756)
  Sets whether the receiver automatically creates and inserts new content objects automatically when loading from a nib file.
- `automaticallyPreparesContent` (page 1747)
  Returns a Boolean value that indicates whether the receiver automatically prepares its content when it is loaded from a nib.
- `prepareContent` (page 1753)
  Typically overridden by subclasses that require additional control over the creation of new objects.

Setting the Content Class

- `setObjectClass:` (page 1758)
  Sets the object class to use when creating new objects.
- `objectClass` (page 1753)
  Returns the class used when creating new non-Core Data objects.

Managing Objects

- `newObject` (page 1752)
  Creates and returns a new object of the appropriate class.
- `addObject:` (page 1746)
  Sets the receiver's content object.
- `removeObject:` (page 1754)
  Removes a given object from the receiver's content.
- `add:` (page 1746)
  Creates a new object and sets it as the receiver's content object.
- `canAdd` (page 1747)
  Returns a Boolean value that indicates whether an object can be added to the receiver using `add:` (page 1746).
- `remove:` (page 1754)
  Removes the receiver's content object.
- `canRemove` (page 1748)
  Returns a Boolean value that indicates whether an object can be removed from the receiver.

Managing Editing

- `setEditable:` (page 1756)
  Sets whether the receiver allows adding and removing objects.

- `isEditable` (page 1751)
  Returns a Boolean value that indicates whether the receiver allows adding and removing objects.

Core Data Support

- `entityName` (page 1749)
  Returns the entity name used by the receiver to create new objects.
- `setEntityName:` (page 1757)
  Sets the entity name used by the receiver to create new objects.
- `fetch:` (page 1749)
  Causes the receiver to fetch the data objects specified by the entity name and fetch predicate.
- `setUsesLazyFetching:` (page 1759)
  Sets whether the receiver uses lazy fetching.
- `usesLazyFetching` (page 1759)
  Returns a Boolean indicating whether the receiver uses lazy fetching.
- `defaultFetchRequest` (page 1749)
  Returns the default fetch request used by the receiver.
- `fetchPredicate` (page 1750)
  Returns the receiver's fetch predicate.
- `setFetchPredicate:` (page 1757)
  Sets the receiver's fetch predicate.
- `managedObjectContext` (page 1751)
  Returns the receiver's managed object context.
- `setManagedObjectContext:` (page 1758)
  Sets the receiver's managed object context.
- `fetchWithRequest:merge:error:` (page 1750)
  Subclasses should override this method to customize a fetch request, for example to specify fetch limits.

Obtaining Selections

- `selectedObjects` (page 1755)
  Returns an array of all objects to be affected by editing.
- `selection` (page 1755)
  Returns a proxy object representing the receiver's selection.

Validating User Interface Items

- `validateUserInterfaceItem:` (page 1759)
  Returns whether the receiver can handle the action method for a user interface item.

Instance Methods

add:

Creates a new object and sets it as the receiver's content object.

```
- (void)add:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

Creates a new object of the appropriate entity (specified by entityName (page 1749)) or class (specified by `objectClass` (page 1753))—see newObject (page 1752)—and sets it as the receiver's content object using `addObject:` (page 1746).

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism can provide feedback as a sheet.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `canAdd` (page 1747)
- `remove:` (page 1754)

Declared In

`NSObjectController.h`

addObject:

Sets the receiver's content object.

```
- (void)addObject:(id)object
```

Parameters

object

The content object for the receiver.

Discussion

If the receiver's content is bound to another object or controller through a relationship key, the relationship of the "master" object is changed.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `removeObject:` (page 1754)

Related Sample Code
Departments and Employees

Declared In
`NSObjectController.h`

automaticallyPreparesContent

Returns a Boolean value that indicates whether the receiver automatically prepares its content when it is loaded from a nib.

```
- (BOOL)automaticallyPreparesContent
```

Return Value
`YES` if the receiver automatically prepares its content when loaded from a nib, otherwise `NO`.

Discussion
See `setAutomaticallyPreparesContent:` (page 1756) for a full explanation of "automatically prepares content."

The default is `NO`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setAutomaticallyPreparesContent:` (page 1756)
- `prepareContent` (page 1753)

Declared In
`NSObjectController.h`

canAdd

Returns a Boolean value that indicates whether an object can be added to the receiver using `add:` (page 1746).

```
- (BOOL)canAdd
```

Return Value
`YES` if an object can be added to the receiver using `add:` (page 1746), otherwise `NO`.

Discussion
Bindings can use this method to control the enabling of user interface objects.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `canRemove` (page 1748)
- `add:` (page 1746)

Declared In
NSObjectController.h

canRemove

Returns a Boolean value that indicates whether an object can be removed from the receiver.

```
- (BOOL)canRemove
```

Return Value
YES if an object can be removed from the receiver using remove: (page 1754), otherwise NO.

Discussion
Bindings can use this method to control the enabling of user interface objects.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- canAdd (page 1747)
- remove: (page 1754)

Declared In
NSObjectController.h

content

Returns the receiver's content object.

```
- (id)content
```

Return Value
The receiver's content object.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setContent: (page 1756)

Related Sample Code
Aperture Edit Plugin - Borders & Titles

Declared In
NSObjectController.h

defaultFetchRequest

Returns the default fetch request used by the receiver.

```
- (NSFetchRequest *)defaultFetchRequest
```

Return Value
The default NSFetchResult used by the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setUsesLazyFetching:` (page 1759)
- `usesLazyFetching` (page 1759)

Declared In
`NSObjectController.h`

entityName

Returns the entity name used by the receiver to create new objects.

```
- (NSString *)entityName
```

Return Value
The entity name used by the receiver to create new objects.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setEntityName:` (page 1757)

Declared In
`NSObjectController.h`

fetch:

Causes the receiver to fetch the data objects specified by the entity name and fetch predicate.

```
- (void)fetch:(id)sender
```

Parameters
sender
Typically the object that invoked this method.

Special Considerations
Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism can provide feedback as a sheet.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setFetchPredicate:` (page 1757)
- `fetchPredicate` (page 1750)

Declared In
`NSObjectController.h`

fetchPredicate

Returns the receiver's fetch predicate.

```
- (NSPredicate *)fetchPredicate
```

Return Value
The receiver's fetch predicate.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `fetch:` (page 1749)
- `setFetchPredicate:` (page 1757)

Declared In
`NSObjectController.h`

fetchWithRequest:merge:error:

Subclasses should override this method to customize a fetch request, for example to specify fetch limits.

```
- (BOOL)fetchWithRequest:(NSFetchRequest *)fetchRequest merge:(BOOL)merge
    error:(NSError **)error
```

Parameters

fetchRequest
The fetch request to use for the fetch. Pass `nil` to use the default fetch request.

merge
If `YES`, the receiver merges the existing content with the fetch result, otherwise the receiver replaces the entire content with the fetch result.

error
If an error occurs, upon return contains an `NSError` object that describes the problem.

Return Value
`YES` if the fetch completed successfully, otherwise `NO`.

Discussion
This method performs a number of actions that you cannot reproduce. To customize this method, you should therefore create your own fetch request and then invoke `super`'s implementation with the new fetch request.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `fetch:` (page 1749)

Declared In
`NSObjectController.h`

initWithContent:

Initializes and returns an `NSObjectController` object with the given content.

```
- (id)initWithContent:(id)content
```

Parameters

content
The content for the receiver.

Return Value
The initialized object controller, with its content object set to *content*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSObjectController.h`

isEditable

Returns a Boolean value that indicates whether the receiver allows adding and removing objects.

```
- (BOOL)isEditable
```

Return Value
`YES` if the receiver allows adding and removing objects, otherwise `NO`.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setEditable:` (page 1756)

Declared In
`NSObjectController.h`

managedObjectContext

Returns the receiver's managed object context.

```
- (NSManagedObjectContext *)managedObjectContext
```

Return Value
The receiver's managed object context.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setManagedObjectContext:` (page 1758)

Related Sample Code
Departments and Employees

Declared In
`NSObjectController.h`

newObject

Creates and returns a new object of the appropriate class.

```
- (id)newObject
```

Return Value
A new object of the appropriate class. The returned object is implicitly retained, the sender is responsible for releasing it (with either release or autorelease).

If an entity name is set (see setEntityName: (page 1757)), the object created is an instance of the class specified for that entity (and the object is inserted into the receiver's managed object context). Otherwise the object created is an instance of the class returned by objectClass (page 1753).

Discussion
This method is called when adding and inserting objects if `automaticallyPreparesContent` (page 1747) is `YES`.

The default implementation assumes the class returned by `objectClass` (page 1753) has a standard `init` method without arguments. If the object class being controlled is `NSManagedObject` (or a subclass thereof) its designated initializer (`initWithEntity:insertIntoManagedObjectContext:`) is called instead, using the entity and managed object context specified for the receiver.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setObjectClass:` (page 1758)
- `objectClass` (page 1753)
- `setEntityName:` (page 1757)
- `entityName` (page 1749)

Related Sample Code
DemoMonkey
Departments and Employees
With and Without Bindings

Declared In
NSObjectController.h

objectClass

Returns the class used when creating new non-Core Data objects.

```
- (Class)objectClass
```

Return Value
The object class used when creating new non-Core Data objects (that is, if no entity has been set)—see newObject (page 1752).

Discussion
If an entity has been set, then the class returned by this method does not automatically reflect the class for the entity.

The default class is NSMutableDictionary.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setObjectClass: (page 1758)
- entityName (page 1749)
- managedObjectContext (page 1751)

Declared In
NSObjectController.h

prepareContent

Typically overridden by subclasses that require additional control over the creation of new objects.

```
- (void)prepareContent
```

Discussion
Subclasses that implement this method are responsible for creating the new content object and setting it as the receiver's content object. This method is only called if automaticallyPreparesContent (page 1747) has been set to YES.

Availability
Available in Mac OS X v10.3 and later.

See Also
- automaticallyPreparesContent (page 1747)
- setAutomaticallyPreparesContent: (page 1756)

Related Sample Code
QTMetadataEditor

Declared In
NSObjectController.h

remove:

Removes the receiver's content object.

```
- (void)remove:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
Removes the receiver's content object using `removeObject:` (page 1754).

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism can provide feedback as a sheet.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `canRemove` (page 1748)
- `add:` (page 1746)

Declared In
NSObjectController.h

removeObject:

Removes a given object from the receiver's content.

```
- (void)removeObject:(id)object
```

Parameters

object
The object to remove from the receiver.

Discussion
If *object* is the receiver's content object, the receiver's content is set to `nil`. If the receiver's content is bound to another object or controller through a relationship key, the relationship of the 'master' object is cleared.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `addObject:` (page 1746)

Declared In
NSObjectController.h

selectedObjects

Returns an array of all objects to be affected by editing.

```
- (NSArray *)selectedObjects
```

Return Value
An array of all objects to be affected by editing. If the receiver supports a selection mechanism, the array contains key value coding compliant proxies of the selected objects; otherwise proxies for all content objects are returned. If the receiver is a concrete instance of `NSObjectController`, returns an array containing the receiver's content object.

Discussion
You should avoid registering for key-value observing changes for key paths that pass *through* this method, (for example, `selectedObjects.firstName`). Using the proxy returned by the `selection` (page 1755) method is better for performance.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- selection` (page 1755)

Declared In
`NSObjectController.h`

selection

Returns a proxy object representing the receiver's selection.

```
- (id)selection
```

Return Value
A proxy object representing the receiver's selection. This object is fully key-value coding compliant, but note that it is a proxy and so does not provide the full range of functionality that might be available in the source object.

Discussion
If a value requested from the selection proxy using key-value coding returns multiple objects, the controller has no selection, or the proxy is not key-value coding compliant for the requested key, the appropriate marker (`NSMultipleValuesMarker`, `NSNoSelectionMarker` or `NSNotApplicableMarker`) is returned. Otherwise, the value of the key is returned.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- selectedObjects` (page 1755)

Declared In
`NSObjectController.h`

setAutomaticallyPreparesContent:

Sets whether the receiver automatically creates and inserts new content objects automatically when loading from a nib file.

```
- (void)setAutomaticallyPreparesContent:(BOOL)flag
```

Parameters

flag

A flag that specifies whether the receiver automatically prepares its content.

Discussion

If *flag* is YES and the receiver is not using a managed object context, prepareContent (page 1753) is used to create the content object. If *flag* is YES and a managed object context is set, the initial content is fetched from the managed object context using the current fetch predicate. The default is NO.

Setting *flag* to YES is the same as checking the "Automatically Prepares Content" option in the Interface Builder controller inspector.

Availability

Available in Mac OS X v10.3 and later.

See Also

- automaticallyPreparesContent (page 1747)
- prepareContent (page 1753)

Declared In

NSObjectController.h

setContent:

Sets the receiver's content object.

```
- (void)setContent:(id)content
```

Parameters

content

The content object for the receiver.

Availability

Available in Mac OS X v10.3 and later.

See Also

- content (page 1748)

Declared In

NSObjectController.h

setEditable:

Sets whether the receiver allows adding and removing objects.

```
- (void)setEditable:(BOOL)flag
```

Parameters

flag

YES if the the receiver should allow adding and removing objects, otherwise NO.

Discussion

The default is YES.

Availability

Available in Mac OS X v10.3 and later.

See Also

- isEditable (page 1751)

Declared In

NSObjectController.h

setEntityName:

Sets the entity name used by the receiver to create new objects.

```
- (void)setEntityName:(NSString *)entityName
```

Parameters

entityName

The entity name used by the receiver to create new objects.

Availability

Available in Mac OS X v10.4 and later.

See Also

- entityName (page 1749)

Declared In

NSObjectController.h

setFetchPredicate:

Sets the receiver's fetch predicate.

```
- (void)setFetchPredicate:(NSPredicate *)predicate
```

Parameters

predicate

The fetch predicate for the receiver.

Discussion

The receiver uses *predicate* when fetching its content, for example in fetch: (page 1749). If you need to customize the fetching behavior further, you can override fetchWithRequest:merge:error: (page 1750).

Availability

Available in Mac OS X v10.4 and later.

See Also

- fetch: (page 1749)

- fetchPredicate (page 1750)
- fetchWithRequest:merge:error: (page 1750)

Declared In
NSObjectController.h

setManagedObjectContext:

Sets the receiver's managed object context.

```
- (void)setManagedObjectContext:(NSManagedObjectContext *)managedObjectContext
```

Parameters

managedObjectContext
The managed object context for the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- managedObjectContext (page 1751)

Declared In
NSObjectController.h

setObjectClass:

Sets the object class to use when creating new objects.

```
- (void)setObjectClass:(Class)objectClass
```

Parameters

objectClass
The object class to use when creating new objects.

Discussion
NSObjectController's default implementation assumes that instances of *objectClass* are initialized using a standard `init` method that takes no arguments.

If an entity name has been set (see setEntityName: (page 1757)), this method has no effect.

Availability
Available in Mac OS X v10.3 and later.

See Also
- objectClass (page 1753)
- setEntityName: (page 1757)
- managedObjectContext (page 1751)

Declared In
NSObjectController.h

setUsesLazyFetching:

Sets whether the receiver uses lazy fetching.

```
- (void)setUsesLazyFetching:(BOOL)enabled
```

Parameters

enabled

Boolean value that indicates whether the receiver uses lazy fetching.

Discussion

When enabled the controller uses a number of techniques that typically make managing large data sets more efficient. As with all optimizations, you should use suitable performance analysis tools (such as Instruments) to determine the best solution.

Note: Setting `setUsesLazyFetching:` to `YES` will cause an exception if the receiving controller is not bound to a managed object context.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `defaultFetchRequest` (page 1749)
- `usesLazyFetching` (page 1759)

Declared In

`NSObjectController.h`

usesLazyFetching

Returns a Boolean indicating whether the receiver uses lazy fetching.

```
- (BOOL)usesLazyFetching
```

Return Value

`YES` if the receiver uses lazy fetching, otherwise `NO`.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `defaultFetchRequest` (page 1749)
- `setUsesLazyFetching:` (page 1759)

Declared In

`NSObjectController.h`

validateUserInterfaceItem:

Returns whether the receiver can handle the action method for a user interface item.

```
- (BOOL)validateUserInterfaceItem:(id < NSValidatedUserInterfaceItem >)item
```

Parameters

item

The user interface item to validate. You can send *item* the `action` (page 3925) and `tag` (page 3926) messages.

Return Value

`YES` if the receiver can handle the action method; `NO` if it cannot.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSObjectController.h`

NSOpenGLContext Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSOpenGL.h
Companion guide	Cocoa Drawing Guide
Related sample code	GLUT LiveVideoMixer2 OpenGLCaptureToMovie Quartz Composer Live DV Quartz Composer Texture

Overview

All OpenGL calls are rendered into an OpenGL graphics context, which in Cocoa is represented by the NSOpenGLContext class. The context is created using an NSOpenGLPixelFormatobject that specifies the context's buffer types and other attributes. A context can be full-screen, offscreen, or associated with an `NSView` object. A context draws into its **drawable object**, which is the frame buffer that is the target of OpenGL drawing operations.

Every NSOpenGLContext object wraps a low-level, platform-specific Core OpenGL (CGL) context. Your application can retrieve the CGL context by calling the `CGLContextObj` (page 1764) method. For more information on the underling CGL context, see *CGL Reference*.

Tasks

Context Creation

`- initWithFormat:shareContext:` (page 1768)
Returns an `NSOpenGLContext` object initialized with the specified pixel format information.

`- initWithCGLContextObj:` (page 1768)
Initializes and returns a `NSOpenGLContext` object using an existing CGL context.

Managing the Current Context

`+ clearCurrentContext` (page 1763)
Sets the current context to `nil`.

`+ currentContext` (page 1764)
Returns the current OpenGL graphics context.

`- makeCurrentContext` (page 1769)
Sets the receiver as the current OpenGL context object.

Drawable Object Management

`- setView:` (page 1775)
Sets the receiver's viewport to the specified `NSView` object.

`- view` (page 1776)
Returns the receiver's view.

`- setFullScreen` (page 1772)
Sets the receiver to full-screen mode.

`- setOffScreen:width:height:rowbytes:` (page 1772)
Instructs the receiver to render into an offscreen buffer with the specified attributes.

`- clearDrawable` (page 1765)
Disassociates the receiver from its viewport.

`- update` (page 1776)
Updates the receiver's drawable object.

Flushing the Drawing Buffer

`- flushBuffer` (page 1767)
Copies the back buffer to the front buffer of the receiver.

Copying Attributes

`- copyAttributesFromContext:withMask:` (page 1765)
Copies selected groups of state variables to the receiver.

Context Parameter Handling

`- setValues:forParameter:` (page 1775)
Sets the value of the specified parameter.

`- getValues:forParameter:` (page 1767)
Returns the value of the requested parameter.

Working with Virtual Screens

- `setCurrentVirtualScreen:` (page 1771)
  Sets the current virtual screen for the receiver.
- `currentVirtualScreen` (page 1766)
  Returns the current virtual screen for the receiver.

Creating Textures

- `createTexture:fromView:internalFormat:` (page 1766)
  Creates a new texture from the contents of the specified view.

Getting the CGL Context Object

- `CGLContextObj` (page 1764)
  Returns the low-level, platform-specific Core OpenGL (CGL) context object represented by the receiver.

Working with Pixel Buffers

- `setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen:` (page 1773)
  Attaches the specified pixel buffer to the receiver.
- `pixelBuffer` (page 1770)
  Returns the pixel-buffer object attached to the receiver.
- `pixelBufferCubeMapFace` (page 1770)
  Returns the cube map face of the pixel buffer attached to the receiver.
- `pixelBufferMipMapLevel` (page 1771)
  Returns the mipmap level of the pixel buffer attached to the receiver.
- `setTextureImageToPixelBuffer:colorBuffer:` (page 1774)
  Attaches the image data in the specified pixel buffer to the texture object currently bound by the receiver.

Class Methods

clearCurrentContext

Sets the current context to `nil`.

```
+ (void)clearCurrentContext
```

Discussion
Until you issue a subsequent call to the `makeCurrentContext` (page 1769) method, OpenGL calls do nothing.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ currentContext` (page 1764)

Related Sample Code
GLUT
NSOpenGL Fullscreen
OpenGLScreenSnapshot

Declared In
`NSOpenGL.h`

currentContext

Returns the current OpenGL graphics context.

```
+ (NSOpenGLContext *)currentContext
```

Return Value
The current OpenGL graphics context, or `nil` if no such object has been set.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ clearCurrentContext` (page 1763)
`+ currentContext` (page 1764)
`- makeCurrentContext` (page 1769)

Related Sample Code
VBL

Declared In
`NSOpenGL.h`

Instance Methods

CGLContextObj

Returns the low-level, platform-specific Core OpenGL (CGL) context object represented by the receiver.

```
- (void *)CGLContextObj
```

Return Value
A pointer to the `CGLContextObj` data type represented by the receiver.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
Denoise

DispatchFractal
OpenGLCaptureToMovie
Quartz Composer Live DV
Quartz Composer Texture

Declared In
NSOpenGL.h

clearDrawable

Disassociates the receiver from its viewport.

```
- (void)clearDrawable
```

Discussion
This method disassociates the receiver from any associated NSView object. If the receiver is in full-screen or offscreen mode, it exits that mode.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setFullScreen (page 1772)
- setOffScreen:width:height:rowbytes: (page 1772)
- setView: (page 1775)
- view (page 1776)

Related Sample Code
GLUT
NSOpenGL Fullscreen

Declared In
NSOpenGL.h

copyAttributesFromContext:withMask:

Copies selected groups of state variables to the receiver.

```
- (void)copyAttributesFromContext:(NSOpenGLContext *)context
    withMask:(GLbitfield)mask
```

Parameters

context
The OpenGL graphics context containing the desired state variables.

mask
A bitfield containing a bitwise OR of the same symbolic names that are passed to the OpenGL call glPushAttrib. The single symbolic constant GL_ALL_ATTRIB_BITS can be used to copy the maximum possible portion of the rendering state.

Discussion
Not all values for OpenGL states can be copied. For example, the pixel pack and unpack state, render mode state, and select and feedback state are not copied. The state that can be copied is exactly the state that is manipulated by the OpenGL call `glPushAttrib`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOpenGL.h`

createTexture:fromView:internalFormat:

Creates a new texture from the contents of the specified view.

```
- (void)createTexture:(GLenum)target fromView:(NSView *)view
    internalFormat:(GLenum)format
```

Parameters

target
The identifier for the new texture.

view
The view to use to generate the texture. This parameter must be either an `NSOpenGLView` object or some other kind of `NSView` object that's associated with an `NSOpenGLContext` object.

format
The format for the texture, interpreted as a `GLenum` data type.

Discussion
The new texture is assigned the identifier in the *target* parameter and is associated with the receiver's context.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
GLUT

Declared In
`NSOpenGL.h`

currentVirtualScreen

Returns the current virtual screen for the receiver.

```
- (GLint)currentVirtualScreen
```

Return Value
The virtual screen number, which is a value between 0 and the number of virtual screens minus one.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `setCurrentVirtualScreen:` (page 1771)

Related Sample Code
Cocoa OpenGL
Quartz Composer Offline Rendering
Quartz Composer Texture

Declared In
`NSOpenGL.h`

flushBuffer

Copies the back buffer to the front buffer of the receiver.

```
- (void)flushBuffer
```

Discussion
If the receiver is not a double-buffered context, this call does nothing.

If the NSOpenGLPixelFormat object used to create the context had a `NO` backing store attribute (`NSOpenGLPFABackingStore`), the buffers may be exchanged rather than copied. This is often the case in full-screen mode.

According to the swap interval context attribute (see `NSOpenGLCPSwapInterval` (page 1777)), the copy may take place during the vertical retrace of the monitor, rather than immediately after `flushBuffer` is called. An implicit `glFlush` is done by `flushBuffer` before it returns. For optimal performance, an application should not call `glFlush` immediately before calling `flushBuffer`. Subsequent OpenGL commands can be issued immediately after calling `flushBuffer`, but are not executed until the buffer copy is completed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `getValues:forParameter:` (page 1767)
- `initWithFormat:shareContext:` (page 1768)
- `setValues:forParameter:` (page 1775)

Related Sample Code
Cocoa OpenGL
DispatchLife
NSOpenGL Fullscreen
NURBSSurfaceVertexProg
SurfaceVertexProgram

Declared In
`NSOpenGL.h`

getValues:forParameter:

Returns the value of the requested parameter.

```
- (void)getValues:(GLint *)vals forParameter:(NSOpenGLContextParameter)param
```

Parameters

vals

On input, a pointer to a variable with enough space for one or more `long` integers. On output, the variable contains the value (or values) for the given parameter.

param

The parameter you want to get. For a list of parameters, see the table in `NSOpenGLContextParameter` (page 1777).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setValues:forParameter:` (page 1775)

Declared In
`NSOpenGL.h`

initWithCGLContextObj:

Initializes and returns a `NSOpenGLContext` object using an existing CGL context.

```
- (id)initWithCGLContextObj:(void *)context
```

Parameters

context

The CGL context to wrap inside the `NSOpenGLContext` object.

Return Value
An initialized context.

Discussion
If your application already has a CGL context, you can wrap a `NSOpenGLContext` object around it using this method. This method retains the CGL context by calling `CGLRetainContext`.

Only one `NSOpenGLContext` object can wrap a specific context.

Your application should not call `CGLDestroyContext` to dispose of the CGL context. Instead, your application should call `CGLReleaseContext` to decrement its reference count.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSOpenGL.h`

initWithFormat:shareContext:

Returns an `NSOpenGLContext` object initialized with the specified pixel format information.

```
- (id)initWithFormat:(NSOpenGLPixelFormat *)format shareContext:(NSOpenGLContext
    *)share
```

Parameters

format

The pixel format to request for the OpenGL graphics context. Following successful initialization, the value you pass in for this parameter is no longer needed and can be deallocated.

share

Another OpenGL graphics context whose texture namespace and display lists you want to share with the receiver. If you do not want to share those features with another graphics context, you may pass `nil` for this parameter.

Return Value

An `NSOpenGLContext` object initialized with the specified parameters, or `nil` if the object could not be created.

Discussion

If the parameters contain invalid information, the receiver releases itself and this method returns `nil`. This may happen if one of the following situations occurs:

- The *format* parameter is `nil` or contains an invalid pixel format.
- The *share* parameter is not `nil` and contains an invalid context.
- The *share* parameter contains a context with a pixel format that is incompatible with the one in *format*.

Pixel formats are incompatible if they use different renderers; this can happen if, for example, one format required an accumulation buffer that could only be provided by the software renderer, and the other format did not.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CubePuzzle

DispatchLife

LiveVideoMixer2

Quartz Composer QCTV

Quartz Composer Texture

Declared In

`NSOpenGL.h`

makeCurrentContext

Sets the receiver as the current OpenGL context object.

```
- (void)makeCurrentContext
```

Discussion

Subsequent OpenGL calls are rendered into the context defined by the receiver.

Note: A context is current on a per-thread basis. Multiple threads must serialize calls into the same context object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ clearCurrentContext` (page 1763)
`+ currentContext` (page 1764)

Related Sample Code
GLUT
LiveVideoMixer2
NURBSSurfaceVertexProg
OpenGL Filter Basics Cocoa
SurfaceVertexProgram

Declared In
`NSOpenGL.h`

pixelBuffer

Returns the pixel-buffer object attached to the receiver.

```
- (NSOpenGLPixelBuffer *)pixelBuffer
```

Return Value
The pixel buffer object.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen:` (page 1773)

Declared In
`NSOpenGL.h`

pixelBufferCubeMapFace

Returns the cube map face of the pixel buffer attached to the receiver.

```
- (GLenum)pixelBufferCubeMapFace
```

Return Value
For pixel buffers with a texture target of `GL_CUBE_MAP`, this value is zero or one of the following values:

- `GL_TEXTURE_CUBE_MAP_POSITIVE_X`
- `GL_TEXTURE_CUBE_MAP_POSITIVE_Y`
- `GL_TEXTURE_CUBE_MAP_POSITIVE_Z`

- GL_TEXTURE_CUBE_MAP_NEGATIVE_X
- GL_TEXTURE_CUBE_MAP_NEGATIVE_Y
- GL_TEXTURE_CUBE_MAP_NEGATIVE_Z

Availability
Available in Mac OS X v10.3 and later.

See Also
- setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen: (page 1773)

Declared In
NSOpenGL.h

pixelBufferMipMapLevel

Returns the mipmap level of the pixel buffer attached to the receiver.

```
- (GLint)pixelBufferMipMapLevel
```

Return Value
The desired mipmap level for rendering. This value should be less than or equal to the maximum texture mipmap level of *pixelBuffer* (accessible through an NSOpenGLPixelBuffer object's textureMaxMipMapLevel (page 1789) method).

Availability
Available in Mac OS X v10.3 and later.

See Also
- setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen: (page 1773)

Declared In
NSOpenGL.h

setCurrentVirtualScreen:

Sets the current virtual screen for the receiver.

```
- (void)setCurrentVirtualScreen:(GLint)screen
```

Parameters
screen
The virtual screen number, which is a value between 0 and the number of virtual screens minus one.

Availability
Available in Mac OS X v10.2 and later.

See Also
- currentVirtualScreen (page 1766)

Declared In
NSOpenGL.h

setFullScreen

Sets the receiver to full-screen mode.

```
- (void)setFullScreen
```

Discussion
In full-screen mode, the receiver renders onto the entire screen. The receiver's viewport is set to the full size of the screen. Call the `clearDrawable` (page 1765) method to exit full-screen mode.

The `NSOpenGLPFAFullScreen` attribute must have been specified in the receiver's NSOpenGLPixelFormat. Some OpenGL renderers, like the software renderer, do not support full-screen mode. The following code determines if a full-screen pixel format is possible on a given system:

```
NSOpenGLPixelFormatAttribute attrs[] =
{
    NSOpenGLPFAFullScreen,
    nil
};

NSOpenGLPixelFormat* pixFmt = [[NSOpenGLPixelFormat alloc]
initWithAttributes:attrs];

/* Check if initWithAttributes succeeded. */
if(pixFmt == nil) {
    /* initWithAttributes failed. There is no full-screen renderer.  */
}
```

Note: It is recommended that an application use Core Graphics's **Direct Display** API to capture the display before entering full-screen mode and release it after exiting. A captured display prevents contention from other applications and system services. In addition, applications are not notified of display changes, preventing them from repositioning their windows and the Finder from repositioning desktop icons.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DispatchLife
GLUT
NSOpenGL Fullscreen
VBL

Declared In
`NSOpenGL.h`

setOffScreen:width:height:rowbytes:

Instructs the receiver to render into an offscreen buffer with the specified attributes.

```
- (void)setOffScreen:(void *)baseaddr width:(GLsizei)width height:(GLsizei)height
    rowbytes:(GLint)rowbytes
```

Parameters

baseaddr

The base address of the buffer in memory. This buffer must contain at least *rowbytes* * *height* bytes.

width

The width of the memory buffer, measured in pixels.

height

The height of the memory buffer, measured in pixels.

rowbytes

The number of bytes in a single row of the buffer. This value must be greater than or equal to the value in *width* times the number of bytes per pixel.

Discussion

The receiver's viewport is set to the full size of the offscreen area. Call the `clearDrawable` (page 1765) method to exit offscreen mode.

The `NSOpenGLPFAOffScreen` attribute must have been specified in the receiver's pixel format object.

Note: To obtain behavior similar to offscreen mode on renderers that do not support accelerated offscreen contexts, attach the context to a hidden window and use `glReadPixels`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSOpenGL.h`

setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen:

Attaches the specified pixel buffer to the receiver.

```
- (void)setPixelBuffer:(NSOpenGLPixelBuffer *)pixelBuffer cubeMapFace:(GLenum)face
    mipMapLevel:(GLint)level currentVirtualScreen:(GLint)screen
```

Parameters

pixelBuffer

The pixel buffer to attach.

face

For pixel buffers with a texture target of `GL_CUBE_MAP`, this parameter should be zero or one of the following values:

- `GL_TEXTURE_CUBE_MAP_POSITIVE_X`
- `GL_TEXTURE_CUBE_MAP_POSITIVE_Y`
- `GL_TEXTURE_CUBE_MAP_POSITIVE_Z`
- `GL_TEXTURE_CUBE_MAP_NEGATIVE_X`
- `GL_TEXTURE_CUBE_MAP_NEGATIVE_Y`
- `GL_TEXTURE_CUBE_MAP_NEGATIVE_Z`

level

The desired mipmap level for rendering. This value must be less than or equal to the maximum texture mipmap level of *pixelBuffer* (accessible through an NSOpenGLPixelBuffer object's textureMaxMipMapLevel (page 1789) method).

screen

The virtual screen of the receiver (if applicable) should be set to the same value as the current virtual screen you are using for rendering onscreen

Discussion
The NSOpenGLPixelBuffer object gives the receiver access to accelerated offscreen rendering in the pixel buffer, which is primarily used for textures.

Availability
Available in Mac OS X v10.3 and later.

See Also
- pixelBufferCubeMapFace (page 1770)
- pixelBufferMipMapLevel (page 1771)
- setCurrentVirtualScreen: (page 1771)
- initWithTextureTarget:textureInternalFormat:textureMaxMipMapLevel:pixelsWide:pixelsHigh: (page 1787) (NSOpenGLPixelBuffer)

Related Sample Code
Quartz Composer Offline Rendering
Quartz Composer Texture

Declared In
NSOpenGL.h

setTextureImageToPixelBuffer:colorBuffer:

Attaches the image data in the specified pixel buffer to the texture object currently bound by the receiver.

```
- (void)setTextureImageToPixelBuffer:(NSOpenGLPixelBuffer *)pixelBuffer
    colorBuffer:(GLenum)source
```

Parameters

pixelBuffer

The pixel buffer to attach.

source

An OpenGL constant indicating which of the pixel buffer's color buffers to use. Potential values for this parameter include GL_FRONT, GL_BACK, and GL_AUX0.

Discussion
This method corresponds to the Core OpenGL method CGLTexImagePBuffer.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
Quartz Composer Texture

Declared In
NSOpenGL.h

setValues:forParameter:

Sets the value of the specified parameter.

```
- (void)setValues:(const GLint *)vals forParameter:(NSOpenGLContextParameter)param
```

Parameters

vals
The new value (or values) for the parameter.

param
The parameter you want to modify. For a list of parameters, see NSOpenGLContextParameter (page 1777).

Availability
Available in Mac OS X v10.0 and later.

See Also
- getValues:forParameter: (page 1767)

Related Sample Code
Cocoa OpenGL
CoreImageGLTextureFBO
GLFullScreen
LiveVideoMixer
LiveVideoMixer2

Declared In
NSOpenGL.h

setView:

Sets the receiver's viewport to the specified NSView object.

```
- (void)setView:(NSView *)view
```

Parameters

view
The view to use for drawing. The full size of the view is used for the viewport.

Availability
Available in Mac OS X v10.0 and later.

See Also
- clearDrawable (page 1765)
- view (page 1776)

Related Sample Code
GLUT
LiveVideoMixer

LiveVideoMixer2
LiveVideoMixer3

Declared In
NSOpenGL.h

update

Updates the receiver's drawable object.

```
- (void)update
```

Discussion
Call this method whenever the receiver's drawable object changes size or location. A multithreaded application must synchronize all threads that access the same drawable object and call `update` for each thread's context serially.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DispatchLife
Quartz Composer Live DV
VideoViewer

Declared In
NSOpenGL.h

view

Returns the receiver's view.

```
- (NSView *)view
```

Return Value
The view, or `nil` if the receiver has no drawable object, is in full-screen mode, or is in offscreen mode.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `clearDrawable` (page 1765)
- `setFullScreen` (page 1772)
- `setOffScreen:width:height:rowbytes:` (page 1772)
- `setView:` (page 1775)

Declared In
NSOpenGL.h

Constants

NSOpenGLContextParameter

The following attribute names are used by `setValues:forParameter:` (page 1775) and `getValues:forParameter:` (page 1767):

```
typedef enum {
    NSOpenGLCPSwapRectangle = 200,
    NSOpenGLCPSwapRectangleEnable = 201,
    NSOpenGLCPRasterizationEnable = 221,
    NSOpenGLCPSwapInterval = 222,
    NSOpenGLCPSurfaceOrder = 235,
    NSOpenGLCPSurfaceOpacity = 236,
    NSOpenGLCPStateValidation = 301
} NSOpenGLContextParameter;
```

Constants

`NSOpenGLCPSwapRectangle`

Sets or gets the swap rectangle.

The swap rectangle is represented as an array of four `long`s: `{x, y, width, height}`.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLCPSwapRectangleEnable`

Enables or disables the swap rectangle in the context's drawable object.

If enabled, the area that is affected by the `flushBuffer` (page 1767) method is restricted to a rectangle specified by the values of `NSOpenGLCPSwapRectangle`. However, the portion of the drawable object that lies outside of the swap rectangle may still be flushed to the screen by a visibility change or other user interface action.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLCPRasterizationEnable`

If disabled, all rasterization of 2D and 3D primitives is disabled.

This state is useful for debugging and to characterize the performance of an OpenGL driver without actually rendering.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLCPSwapInterval`

Sets or gets the swap interval.

The swap interval is represented as one `long`. If the swap interval is set to `0` (the default), the `flushBuffer` (page 1767) method executes as soon as possible, without regard to the vertical refresh rate of the monitor. If the swap interval is set to `1`, the buffers are swapped only during the vertical retrace of the monitor.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLCPSurfaceOrder`

Get or set the surface order.

If the surface order is set to 1 (the default), the order is above the window (default). If the value is –1, the order is below the window.

Available in Mac OS X v10.2 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLCPSurfaceOpacity`

Set or get the surface opacity.

If the opacity is set to 1 (the default), the surface is opaque. If the value is 0, the surface is non-opaque.

Available in Mac OS X v10.2 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLCPStateValidation`

If enabled, OpenGL inspects the context state each time the `update` (page 1776) method is called to ensure that it is in an appropriate state for switching between renderers.

Normally, the state is inspected only when it is actually necessary to switch renderers. This is useful when using a single monitor system to test that an application performs correctly on a multiple-monitor system.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOpenGL.h`

NSOpenGLLayer Class Reference

Inherits from	CAOpenGLLayer : CALayer : NSObject
Conforms to	NSCoding (CALayer) CAMediaTiming (CALayer) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSOpenGLLayer.h
Companion guide	Core Animation Programming Guide

Overview

`NSOpenGLLayer` is a subclass of `CAOpenGLLayer` that is suitable for rendering OpenGL into layers. Unlike `CAOpenGLLayer`, `NSOpenGLLayer` uses distinctly Application Kit types.

Tasks

Drawing the Content

`- canDrawInOpenGLContext:pixelFormat:forLayerTime:displayTime:` (page 1781)
Invoked to ask the layer whether it can (or should) draw.

`- drawInOpenGLContext:pixelFormat:forLayerTime:displayTime:` (page 1782)
Draws the OpenGL content for the specified time.

Managing the Pixel Format

`openGLPixelFormat` (page 1780) *property*
Provides access to the layer's associated NSOpenGLPixelFormat.

`- openGLPixelFormatForDisplayMask:` (page 1782)
Returns the OpenGL pixel format suitable for the specified displays.

Managing the Rendering Context

`openGLContext` (page 1780) *property*
The layer's NSOpenGLContext.

`- openGLContextForPixelFormat:` (page 1782)
Returns the OpenGL context to use for the requested pixel format.

Accessing the Associated View

`view` (page 1781) *property*
Returns the view associated with the layer.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

openGLContext

The layer's NSOpenGLContext.

```
@property(retain) NSOpenGLContext *openGLContext
```

Discussion
Provides access to the layer's associated `NSOpenGLContext`. Subclasses shouldn't invoke `setOpenGLContext:`, but can override it if desired to intercept assignment of the layer's context.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSOpenGLLayer.h`

openGLPixelFormat

Provides access to the layer's associated NSOpenGLPixelFormat.

```
@property(retain) NSOpenGLPixelFormat *openGLPixelFormat
```

Discussion
Subclasses shouldn't invoke `setOpenGLPixelFormat:`, but can override it if desired to intercept assignment of the layer's pixel format.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSOpenGLLayer.h`

view

Returns the view associated with the layer.

```
@property(assign) NSView *view
```

Discussion
Subclasses shouldn't invoke setView:, but can override it if desired to intercept the layer's association to, or dissociation from, a view.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSOpenGLLayer.h`

Instance Methods

canDrawInOpenGLContext:pixelFormat:forLayerTime:displayTime:

Invoked to ask the layer whether it can (or should) draw.

```
- (BOOL)canDrawInOpenGLContext:(NSOpenGLContext *)context
    pixelFormat:(NSOpenGLPixelFormat *)pixelFormat
    forLayerTime:(CFTimeInterval)timeInterval displayTime:(const CVTimeStamp
    *)timeStamp
```

Parameters

context
The NSOpenGLContext in to which the OpenGL content would be drawn.

pixelFormat
The pixel format used when the context was created.

timeInterval
The current layer time.

timeStamp
The display timestamp associated with timeInterval. Can be `null`.

Return Value
`YES` if the receiver should render OpenGL content, `NO` otherwise.

Discussion
This method is called before attempting to render the frame for the layer time specified by *timeInterval*. If the method returns `NO`, the frame is skipped. The default implementation always returns `YES`.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSOpenGLLayer.h`

drawInOpenGLContext:pixelFormat:forLayerTime:displayTime:

Draws the OpenGL content for the specified time.

```
- (void)drawInOpenGLContext:(NSOpenGLContext *)context
    pixelFormat:(NSOpenGLPixelFormat *)pixelFormat
    forLayerTime:(CFTimeInterval)timeInterval displayTime:(const CVTimeStamp
    *)timeStamp
```

Parameters

context

The NSOpenGLContext in to which the OpenGL content would be drawn.

pixelFormat

The pixel format used when the context was created.

timeInterval

The current layer time.

timeStamp

The display timestamp associated with timeInterval. Can be `null`.

Discussion
This method is called when a new frame needs to be generated for the layer time specified by timeInterval.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSOpenGLLayer.h`

openGLContextForPixelFormat:

Returns the OpenGL context to use for the requested pixel format.

```
- (NSOpenGLContext *)openGLContextForPixelFormat:(NSOpenGLPixelFormat *)pixelFormat
```

Parameters

pixelFormat

The pixel format.

Return Value
An autoreleased `NSOpenGLContext`.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSOpenGLLayer.h`

openGLPixelFormatForDisplayMask:

Returns the OpenGL pixel format suitable for the specified displays.

```
- (NSOpenGLPixelFormat *)openGLPixelFormatForDisplayMask:(uint32_t)mask
```

Parameters

mask

A mask specifying the displays the returned `NSOpenGLPixelFormat` must be suitable for.

Return Value

An autoreleased `NSOpenGLPixelFormat` object suitable for the displays.

Discussion

You must include an NSOpenGLPFAScreenMask specification in the pixel format attribute list that's used to instantiate the NSOpenGLPixelFormat.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSOpenGLLayer.h`

NSOpenGLPixelBuffer Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSOpenGL.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Cocoa Drawing Guide
Related sample code	Quartz Composer Offline Rendering Quartz Composer Texture

Overview

The `NSOpenGLPixelBuffer` class gives Cocoa OpenGL implementations access to accelerated offscreen rendering. With this offscreen rendering you could, for instance, draw into the pixel buffer, then use the contents as a texture map elsewhere. Typically you initialize an `NSOpenGLPixelBuffer` object using the `initWithTextureTarget:textureInternalFormat:textureMaxMipMapLevel:pixelsWide:pixelsHigh:` (page 1787) method and attach the resulting object to an OpenGL context with the `setPixelBuffer:cubeMapFace:mipMapLevel:currentVirtualScreen:` (page 1773) method of `NSOpenGLContext`.

Every `NSOpenGLPixelBuffer` object wraps a low-level, platform-specific Core OpenGL (CGL) pixel buffer object. Your application can retrieve the CGL pixel buffer by calling the `CGLPBufferObj` (page 1786) method. For more information on the underling CGL pixel buffer, see *CGL Reference*.

Tasks

Initializing an OpenGL Pixel Buffer

`- initWithTextureTarget:textureInternalFormat:textureMaxMipMapLevel:pixelsWide:pixelsHigh:` (page 1787)

Returns an `NSOpenGLPixelBuffer` object initialized with the specified parameters.

- `initWithCGLPBufferObj:` (page 1786)
  Initializes and returns an `NSOpenGLPixelBuffer` object that encapsulates an existing CGL pixel buffer object.

Obtaining Information About an OpenGL Pixel Buffer

- `CGLPBufferObj` (page 1786)
  Returns the underlying `CGLPBufferObj` object associated with the `NSOpenGLPixelBuffer` object.
- `pixelsHigh` (page 1788)
  Returns the height of the receiver's texture (in pixels).
- `pixelsWide` (page 1788)
  Returns the width of the receiver's texture (in pixels).
- `textureInternalFormat` (page 1789)
  Returns the internal format of the receiver's texture.
- `textureMaxMipMapLevel` (page 1789)
  Returns the maximum mipmap level of the receiver's texture.
- `textureTarget` (page 1789)
  Returns the texture target of the receiver.

Instance Methods

CGLPBufferObj

Returns the underlying `CGLPBufferObj` object associated with the `NSOpenGLPixelBuffer` object.

```
- (void *)CGLPBufferObj
```

Return Value
The CGL pixel buffer object that encapsulates the actual pixel buffer.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSOpenGL.h`

initWithCGLPBufferObj:

Initializes and returns an `NSOpenGLPixelBuffer` object that encapsulates an existing CGL pixel buffer object.

```
- (id)initWithCGLPBufferObj:(void *)pbuffer
```

Parameters
pbuffer
The CGL pixel buffer object to wrap.

Return Value
An initialized `NSOpenGLPixelBuffer` object.

Discussion
If your application already has a CGL pixel buffer object, you can wrap it inside an `NSOpenGLPixelBuffer` object by using this initializer. This method retains the CGL pixel buffer object by calling the `CGLRetainPBuffer` function.

Your application should not call `CGLDestroyPBuffer` to dispose of the CGL pixel buffer object. Instead, your application should call `CGLReleasePBuffer` to decrement its reference count.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSOpenGL.h`

initWithTextureTarget:textureInternalFormat:textureMaxMipMapLevel:pixelsWide:pixelsHigh:

Returns an `NSOpenGLPixelBuffer` object initialized with the specified parameters.

```
- (id)initWithTextureTarget:(GLenum)target textureInternalFormat:(GLenum)format
    textureMaxMipMapLevel:(GLint)maxLevel pixelsWide:(GLsizei)pixelsWide
    pixelsHigh:(GLsizei)pixelsHigh
```

Parameters

target
The texture object. This value should be one of the following: `GL_TEXTURE_2D`,`GL_TEXTURE_CUBE_MAP`, or `GL_TEXTURE_RECTANGLE_EXT`.

format
The base internal format of the texture. This value should be `GL_RGB`, `GL_RGBA`, or `GL_DEPTH_COMPONENT`.

maxLevel
The desired maximum mipmap level of the structure, starting with zero.

pixelsWide
The width of the texture (in pixels) in the pixel buffer.

pixelsHigh
The height of the texture (in pixels) in the pixel buffer.

Return Value
An initialized `NSOpenGLPixelBuffer` object or `nil` if the initialization failed. Initialization can fail if there is inconsistency among the parameter values. See the OpenGL documentation for `glTexImage2D` for more information.

Discussion
The value you pass to the *target* parameter defines several other constraints that are then applied to the remaining parameters. The list below gives the values you can pass to *target* and the additional constraints.

- `GL_TEXTURE_2D`
- `GL_TEXTURE_CUBE_MAP` - the values in *pixelsWide* and *pixelsHigh* must be equal.

- `GL_TEXTURE_RECTANGLE_EXT` - *`maxLevel`* must be zero.

Normally, when using the `GL_TEXTURE_2D` and `GL_TEXTURE_CUBE_MAP` targets, you must specify width and height values that are powers of two. When the `ARB_texture_non_power_of_two` extension is present, however, some types of hardware can support values that are not powers of two. You should check for the presence of this extension before specifying non power-of-two values.

If the texture map cannot be created, you can use the `glGetError` function to get the error code.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
Quartz Composer Offline Rendering
Quartz Composer Texture

Declared In
`NSOpenGL.h`

pixelsHigh

Returns the height of the receiver's texture (in pixels).

```
- (GLsizei)pixelsHigh
```

Return Value
The height of the texture (in pixels).

Availability
Available in Mac OS X v10.3 and later.

See Also
`- pixelsWide` (page 1788)

Related Sample Code
Quartz Composer Offline Rendering
Quartz Composer Texture

Declared In
`NSOpenGL.h`

pixelsWide

Returns the width of the receiver's texture (in pixels).

```
- (GLsizei)pixelsWide
```

Return Value
The width of the texture (in pixels).

Availability
Available in Mac OS X v10.3 and later.

See Also
- pixelsHigh (page 1788)

Related Sample Code
Quartz Composer Offline Rendering
Quartz Composer Texture

Declared In
NSOpenGL.h

textureInternalFormat

Returns the internal format of the receiver's texture.

```
- (GLenum)textureInternalFormat
```

Return Value
The texture format, which can be one of the following values: GL_RGB, GL_RGBA, or GL_DEPTH_COMPONENT.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSOpenGL.h

textureMaxMipMapLevel

Returns the maximum mipmap level of the receiver's texture.

```
- (GLint)textureMaxMipMapLevel
```

Return Value
The maximum mipmap level.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSOpenGL.h

textureTarget

Returns the texture target of the receiver.

```
- (GLenum)textureTarget
```

Return Value
The texture target, which can be one of the following values: GL_TEXTURE_2D, GL_TEXTURE_CUBE_MAP, or GL_TEXTURE_RECTANGLE_EXT.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSOpenGL.h

NSOpenGLPixelFormat Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSOpenGL.h
Companion guide	Cocoa Drawing Guide
Related sample code	Denoise From A View to A Movie From A View to A Picture GLUT QTCoreVideo201

Overview

To render with OpenGL into an `NSOpenGLContext`, you must specify the context's pixel format. An `NSOpenGLPixelFormat` object specifies the types of buffers and other attributes of the `NSOpenGLContext`. This class is similar to the `AGLPixelFormat` type, which is used in Carbon OpenGL applications.

Every `NSOpenGLPixelFormat` object wraps a low-level, platform-specific Core OpenGL (CGL) pixel format object. Your application can retrieve the CGL pixel format object by calling the `CGLPixelFormatObj` (page 1792) method. For more information on the underling CGL pixel format object, see *CGL Reference*.

Tasks

Creating an NSOpenGLPixelFormat Object

- `initWithCGLPixelFormatObj:` (page 1795)
  Returns an `NSOpenGLPixelFormat`object initialized with using an existing CGL pixel format object.
- `initWithData:` (page 1796)
  Returns an `NSOpenGLPixelFormat` object initialized with specified pixel format attribute data. (Deprecated. Use `initWithAttributes:` (page 1793) instead.)

- `initWithAttributes:` (page 1793)
  Returns an `NSOpenGLPixelFormat` object initialized with specified pixel format attributes.

Managing the Pixel Format

- `CGLPixelFormatObj` (page 1792)
  Returns the low-level, platform-specific Core OpenGL (CGL) pixel format object represented by the receiver.
- `getValues:forAttribute:forVirtualScreen:` (page 1793)
  Gets the value for the specified pixel format attribute.
- `numberOfVirtualScreens` (page 1796)
  Returns the number of virtual screens associated with the receiver.

Managing Attributes

- `attributes` (page 1792)
  Retrieves the attribute data for the pixel format object. (Deprecated.)
- `setAttributes:` (page 1797)
  Sets the attribute data for the pixel format object. (Deprecated.)

Instance Methods

attributes

Retrieves the attribute data for the pixel format object. (Deprecated.)

```
- (NSData *)attributes
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOpenGL.h`

CGLPixelFormatObj

Returns the low-level, platform-specific Core OpenGL (CGL) pixel format object represented by the receiver.

```
- (void *)CGLPixelFormatObj
```

Return Value
A pointer to the underlying `CGLPixelFormatObj` object.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
CIRAWFilterSample
Denoise
DispatchFractal
WebKitCIPlugIn
WhackedTV

Declared In
NSOpenGL.h

getValues:forAttribute:forVirtualScreen:

Gets the value for the specified pixel format attribute.

```
- (void)getValues:(GLint *)vals forAttribute:(NSOpenGLPixelFormatAttribute)attrib
    forVirtualScreen:(GLint)screen
```

Parameters

vals
On input, a pointer to a `long` variable. On output, the variable contains the value of the requested attribute.

attrib
The requested attribute. For a list of attribute constants, see the table in "Constants" (page 1797).

screen
The screen from which you want to retrieve the attribute. This parameter must be a value between 0 and the number of virtual screens (`numberOfVirtualScreens` (page 1796)) minus 1.

Discussion
Because the value for an attribute may be different on each virtual screen, the virtual screen must be specified along with the attribute.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- initWithAttributes:` (page 1793)

Related Sample Code
CIVideoDemoGL
Cocoa OpenGL
GLUT
LiveVideoMixer3
NSOpenGL Fullscreen

Declared In
NSOpenGL.h

initWithAttributes:

Returns an `NSOpenGLPixelFormat` object initialized with specified pixel format attributes.

```
- (id)initWithAttributes:(const NSOpenGLPixelFormatAttribute *)attribs
```

Parameters

attribs

A 0-terminated array containing Boolean and integer attribute constants. The presence of a Boolean attribute implies a value of `YES` while its absence implies a value of `NO`. Integer constants must be followed by the desired value. For a listing of attribute constants, see the table in "Constants" (page 1797).

Return Value

An initialized `NSOpenGLPixelFormat` object whose attributes match the desired attributes as close as possible, or `nil` if an object with the desired attributes could not be initialized.

Discussion

On return, the Boolean attributes of the receiver match the values specified in *attribs*, and the integer attributes are as close to the specified values as can be provided by the system. However, if no matching pixel format exists, the receiver releases itself and `nil` is returned. You may deallocate the receiver following its use in the successful initialization of an `NSOpenGLContext`.

The existence of a Boolean attribute constant in *attribs* implies a `YES` value. The Boolean attribute constants are:

```
NSOpenGLPFAAllRenderers
NSOpenGLPFADoubleBuffer
NSOpenGLPFAStereo
NSOpenGLPFAMinimumPolicy
NSOpenGLPFAMaximumPolicy
NSOpenGLPFAOffScreen
NSOpenGLPFAFullScreen
NSOpenGLPFASingleRenderer
NSOpenGLPFANoRecovery
NSOpenGLPFAAccelerated
NSOpenGLPFAClosestPolicy
NSOpenGLPFARobust
NSOpenGLPFABackingStore
NSOpenGLPFAWindow
NSOpenGLPFAMultiScreen
NSOpenGLPFACompliant
NSOpenGLPFAPixelBuffer
```

The integer constants must be followed by a value. These constants are:

```
NSOpenGLPFAAuxBuffers
NSOpenGLPFAColorSize
NSOpenGLPFAAlphaSize
NSOpenGLPFADepthSize
NSOpenGLPFAStencilSize
NSOpenGLPFAAccumSize
NSOpenGLPFARendererID
NSOpenGLPFAScreenMask
```

This code fragment creates a double-buffered pixel format with a 32-bit depth buffer:

```
NSOpenGLPixelFormatAttribute attrs[] =
{
    NSOpenGLPFADoubleBuffer,
    NSOpenGLPFADepthSize, 32,
    0
};

NSOpenGLPixelFormat* pixFmt = [[NSOpenGLPixelFormat alloc]
initWithAttributes:attrs];

/* Check if initWithAttributes succeeded. */
if(pixFmt == nil) {
    /* initWithAttributes failed. Try to alloc/init with a different  list of
attributes. */
}
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- getValues:forAttribute:forVirtualScreen:` (page 1793)

Related Sample Code
CoreImageGLTextureFBO
DispatchLife
From A View to A Movie
From A View to A Picture
SurfaceVertexProgram

Declared In
`NSOpenGL.h`

initWithCGLPixelFormatObj:

Returns an `NSOpenGLPixelFormat`object initialized with using an existing CGL pixel format object.

```
- (id)initWithCGLPixelFormatObj:(void *)format
```

Parameters

format
An existing CGL pixel format object.

Return Value
An intialized `NSOpenGLPixelFormat` object that wraps the CGL pixel format object.

Discussion
If your application already has a low-level CGL pixel format object, you can create an `NSOpenGLPixelFormat` object to wrap it by calling this initializer. The `NSOpenGLPixelFormat` object retains the CGL pixel format object by calling the `CGLRetainPixelFormat` function.

Your application should not call `CGLDestroyPixelFormat` to dispose of the CGL pixel format object. Instead, your application should call `CGLReleasePixelFormat` to decrement its reference count.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSOpenGL.h

initWithData:

Returns an NSOpenGLPixelFormat object initialized with specified pixel format attribute data. (Deprecated. Use initWithAttributes: (page 1793) instead.)

```
- (id)initWithData:(NSData *)attribs
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSOpenGL.h

numberOfVirtualScreens

Returns the number of virtual screens associated with the receiver.

```
- (GLint)numberOfVirtualScreens
```

Return Value
The number of virtual screens.

Discussion
When the attributes are set, OpenGL searches for drivers matching the requested attributes. Each matching driver drives a set of displays. For example, a graphics card in a portable computer might drive the internal screen and an external display. This portable computer would have one virtual screen. A desktop computer might have two different graphics cards, each driving one or more displays. The pairing of an OpenGL driver with its set of associated displays corresponds to one virtual screen. In the above examples, the portable computer would have one virtual screen, while the desktop computer would have two. Another desktop computer with a video card driving two displays at once would have one virtual screen.

For more information on virtual screens, consult *OpenGL Programming Guide for Mac OS X*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- getValues:forAttribute:forVirtualScreen: (page 1793)

Related Sample Code
CIVideoDemoGL
DispatchFractal
LiveVideoMixer
LiveVideoMixer2
LiveVideoMixer3

Declared In
NSOpenGL.h

setAttributes:

Sets the attribute data for the pixel format object. (Deprecated.)

```
- (void)setAttributes:(NSData *)attribs
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSOpenGL.h

Constants

NSOpenGLPixelFormatAttribute

The following attribute names are used by `initWithAttributes:` (page 1793) and `getValues:forAttribute:forVirtualScreen:` (page 1793):

```
enum {
    NSOpenGLPFAAllRenderers       =   1,
    NSOpenGLPFADoubleBuffer       =   5,
    NSOpenGLPFAStereo             =   6,
    NSOpenGLPFAAuxBuffers         =   7,
    NSOpenGLPFAColorSize          =   8,
    NSOpenGLPFAAlphaSize          =  11,
    NSOpenGLPFADepthSize          =  12,
    NSOpenGLPFAStencilSize        =  13,
    NSOpenGLPFAAccumSize          =  14,
    NSOpenGLPFAMinimumPolicy      =  51,
    NSOpenGLPFAMaximumPolicy      =  52,
    NSOpenGLPFAOffScreen          =  53,
    NSOpenGLPFAFullScreen         =  54,
    NSOpenGLPFASampleBuffers      =  55,
    NSOpenGLPFASamples            =  56,
    NSOpenGLPFAAuxDepthStencil    =  57,
    NSOpenGLPFAColorFloat         =  58,
    NSOpenGLPFAMultisample        =  59,
    NSOpenGLPFASupersample        =  60,
    NSOpenGLPFASampleAlpha        =  61,
    NSOpenGLPFARendererID         =  70,
    NSOpenGLPFASingleRenderer     =  71,
    NSOpenGLPFANoRecovery         =  72,
    NSOpenGLPFAAccelerated        =  73,
    NSOpenGLPFAClosestPolicy      =  74,
    NSOpenGLPFARobust             =  75,
    NSOpenGLPFABackingStore       =  76,
    NSOpenGLPFAMPSafe             =  78,
    NSOpenGLPFAWindow             =  80,
    NSOpenGLPFAMultiScreen        =  81,
    NSOpenGLPFACompliant          =  83,
    NSOpenGLPFAScreenMask         =  84,
    NSOpenGLPFAPixelBuffer        =  90,
    NSOpenGLPFARemotePixelBuffer  =  91,
    NSOpenGLPFAAllowOfflineRenderers = 96,
    NSOpenGLPFAAcceleratedCompute =  97,
    NSOpenGLPFAVirtualScreenCount = 128
};
typedef  uint32_t NSOpenGLPixelFormatAttribute;
```

Constants

`NSOpenGLPFAAllRenderers`

A Boolean attribute. If present, this attribute indicates that the pixel format selection is open to all available renderers, including debug and special-purpose renderers that are not OpenGL compliant.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFADoubleBuffer`

A Boolean attribute. If present, this attribute indicates that only double-buffered pixel formats are considered. Otherwise, only single-buffered pixel formats are considered.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAStereo`
A Boolean attribute. If present, this attribute indicates that only stereo pixel formats are considered. Otherwise, only monoscopic pixel formats are considered.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAAuxBuffers`
Value is a nonnegative integer that indicates the desired number of auxiliary buffers. Pixel formats with the smallest number of auxiliary buffers that meets or exceeds the specified number are preferred.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAColorSize`
Value is a nonnegative buffer size specification. A color buffer that most closely matches the specified size is preferred. If unspecified, OpenGL chooses a color size that matches the screen.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAAlphaSize`
Value is a nonnegative buffer size specification. An alpha buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFADepthSize`
Value is a nonnegative depth buffer size specification. A depth buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAStencilSize`
Value is a nonnegative integer that indicates the desired number of stencil bitplanes. The smallest stencil buffer of at least the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAAccumSize`
Value is a nonnegative buffer size specification. An accumulation buffer that most closely matches the specified size is preferred.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAMinimumPolicy`
A Boolean attribute. If present, this attribute indicates that the pixel format choosing policy is altered for the color, depth, and accumulation buffers such that only buffers of size greater than or equal to the desired size are considered.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAMaximumPolicy`

A Boolean attribute. If present, this attribute indicates that the pixel format choosing policy is altered for the color, depth, and accumulation buffers such that, if a nonzero buffer size is requested, the largest available buffer is preferred.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAOffScreen`

A Boolean attribute. If present, this attribute indicates that only renderers that are capable of rendering to an offscreen memory area and have buffer depth exactly equal to the desired buffer depth are considered. The `NSOpenGLPFAClosestPolicy` attribute is implied.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAFullScreen`

A Boolean attribute. If present, this attribute indicates that only renderers that are capable of rendering to a full-screen drawable are considered. The `NSOpenGLPFASingleRenderer` attribute is implied.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFASampleBuffers`

Value is a nonnegative number indicating the number of multisample buffers.

Available in Mac OS X v10.2 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFASamples`

Value is a nonnegative indicating the number of samples per multisample buffer.

Available in Mac OS X v10.2 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAAuxDepthStencil`

Each auxiliary buffer has its own depth stencil.

Available in Mac OS X v10.2 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAColorFloat`

A Boolean attribute. If present, this attribute indicates that only renderers that are capable using buffers storing floating point pixels are considered. This should be accompanied by a `NSOpenGLPFAColorSize` of 64 (for half float pixel components) or 128 (for full float pixel components). Note, not all hardware supports floating point color buffers thus the returned pixel format could be NULL.

Available in Mac OS X v10.4 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAMultisample`
A Boolean attribute. If present and used with `NSOpenGLPFASampleBuffers` and `NSOpenGLPFASamples`, this attribute hints to OpenGL to prefer multi-sampling. Multi-sampling will sample textures at the back buffer dimensions vice the multi-sample buffer dimensions and use that single sample for all fragments with coverage on the back buffer location. This means less total texture samples than with super-sampling (by a factor of the number of samples requested) and will likely be faster though less accurate (texture sample wise) than super-sampling. If the underlying video card does not have enough VRAM to support this feature, this hint does nothing.

The `NSOpenGLPFASampleBuffers` and `NSOpenGLPFASamples` attributes must be configured to request anti-aliasing as follows:

```
NSOpenGLPFAMultisample,
NSOpenGLPFASampleBuffers, (NSOpenGLPixelFormatAttribute)1
NSOpenGLPFASamples, (NSOpenGLPixelFormatAttribute)4,
```

If after adding these options, multisamping still does not work, try removing the `NSOpenGLPFAPixelBuffer` attribute (if present). Some graphics cards may not support this option in specific versions of Mac OS X. If removing the attribute still does not enable multisampling, try adding the `NSOpenGLPFANoRecovery` attribute.

Available in Mac OS X v10.4 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFASupersample`
A Boolean attribute. If present and used with `NSOpenGLPFASampleBuffers` and `NSOpenGLPFASamples`, this attribute hints to OpenGL to prefer super-sampling. Super-sampling will process fragments with a texture sample per fragment and would likely be slower than multi-sampling. If the pixel format is not requesting anti-aliasing, this hint does nothing.

Available in Mac OS X v10.4 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFASampleAlpha`
A Boolean attribute. If present and used with `NSOpenGLPFASampleBuffers` and `NSOpenGLPFASampleBuffers`, this attribute hints to OpenGL to update multi-sample alpha values to ensure the most accurate rendering. If pixel format is not requesting anti-aliasing then this hint does nothing.

Available in Mac OS X v10.4 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFARendererID`
Value is a nonnegative renderer ID number. OpenGL renderers that match the specified ID are preferred. Constants to select specific renderers are provided in the `CGLRenderers.h` header of the OpenGL framework. Of note is `kCGLRendererGenericID` which selects the Apple software renderer. The other constants select renderers for specific hardware vendors.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFASingleRenderer`
A Boolean attribute. If present, this attribute indicates that a single rendering engine is chosen. On systems with multiple screens, this disables OpenGL's ability to drive different monitors through different graphics accelerator cards with a single context. This attribute is not generally useful.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFANoRecovery`

A Boolean attribute. If present, this attribute indicates that OpenGL's failure recovery mechanisms are disabled. Normally, if an accelerated renderer fails due to lack of resources, OpenGL automatically switches to another renderer. This attribute disables these features so that rendering is always performed by the chosen renderer. This attribute is not generally useful.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAAccelerated`

A Boolean attribute. If present, this attribute indicates that only hardware-accelerated renderers are considered. If not present, accelerated renderers are still preferred.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAClosestPolicy`

A Boolean attribute. If present, this attribute indicates that the pixel format choosing policy is altered for the color buffer such that the buffer closest to the requested size is preferred, regardless of the actual color buffer depth of the supported graphics device.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFARobust`

A Boolean attribute. If present, this attribute indicates that only renderers that do not have any failure modes associated with a lack of video card resources are considered. This attribute is not generally useful.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFABackingStore`

A Boolean attribute. If present, this attribute indicates that OpenGL only considers renderers that have a back color buffer the full size of the drawable (regardless of window visibility) and that guarantee the back buffer contents to be valid after a call to `NSOpenGLContext` object's `flushBuffer` (page 1767).

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAMPSafe`

A Boolean attribute. If present, this attribute indicates that the renderer is multi-processor safe.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAWindow`

A Boolean attribute. If present, this attribute indicates that only renderers that are capable of rendering to a window are considered. This attribute is implied if neither `NSOpenGLPFAFullScreen` nor `NSOpenGLPFAOffScreen` is specified.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAMultiScreen`
A Boolean attribute. If present, this attribute indicates that only renderers capable of driving multiple screens are considered. This attribute is not generally useful.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFACompliant`
A Boolean attribute. If present, this attribute indicates that pixel format selection is only open to OpenGL-compliant renderers. This attribute is implied unless `NSOpenGLPFAAllRenderers` is specified. This attribute is not useful in the attribute array.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAScreenMask`
Value is a bit mask of supported physical screens. All screens specified in the bit mask are guaranteed to be supported by the pixel format. Screens not specified in the bit mask may still be supported. The bit mask is managed by the CoreGraphics's **DirectDisplay**, available in the `CGDirectDisplay.h` header of the ApplicationServices umbrella framework. A `CGDirectDisplayID` must be converted to an OpenGL display mask using the function `CGDisplayIDToOpenGLDisplayMask`. This attribute is not generally useful.

Available in Mac OS X v10.0 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAPixelBuffer`
A Boolean attribute. If present, this attribute indicates that rendering to a pixel buffer is enabled.

Available in Mac OS X v10.3 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFARemotePixelBuffer`
A Boolean attribute. If present, this attribute indicates that rendering to a pixel buffer on an offline renderer is enabled.

Available in Mac OS X v10.6 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAAllowOfflineRenderers`
A Boolean attribute. If present, this attribute indicates that offline renderers may be used.

Available in Mac OS X v10.5 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAAcceleratedCompute`
If present, this attribute indicates that only renderers that can execute OpenCL programs should be used.

Available in Mac OS X v10.6 and later.

Declared in `NSOpenGL.h`.

`NSOpenGLPFAVirtualScreenCount`
The number of virtual screens in this format.

Available in Mac OS X v10.2 and later.

Declared in `NSOpenGL.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOpenGL.h`

NSOpenGLView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSOpenGLView.h
Companion guide	Cocoa Drawing Guide
Related sample code	CIColorTracking From A View to A Movie From A View to A Picture LiveVideoMixer2 LiveVideoMixer3

Overview

An `NSOpenGLView` object maintains an NSOpenGLPixelFormat and NSOpenGLContext object into which OpenGL calls can be rendered. The view provides methods for accessing and managing the `NSOpenGLPixelFormat` and `NSOpenGLContext` objects, as well as notifications of visible region changes.

An `NSOpenGLView` object cannot have subviews. You can, however, divide a single `NSOpenGLView` into multiple rendering areas using the `glViewport` function.

When creating an `NSOpenGLView` object in Interface Builder, you use the inspector window to specify the pixel format attributes you want for the view. Only those attributes listed in the Interface Builder inspector are set when the view is instantiated.

Note: In versions of the Xcode Tools that shipped prior to Mac OS X v10.4, the Interface Builder inspector does not list any pixel format attributes for `NSOpenGLView`.

Tasks

Initializing an NSOpenGLView

`- initWithFrame:pixelFormat:` (page 1808)
Returns an `NSOpenGLView` object initialized with the specified frame rectangle and pixel format.

Managing the NSOpenGLPixelFormat

`+ defaultPixelFormat` (page 1807)
Returns a default NSOpenGLPixelFormat object.

`- pixelFormat` (page 1809)
Returns the NSOpenGLPixelFormat object associated with the receiver.

`- setPixelFormat:` (page 1811)
Sets the receiver's NSOpenGLPixelFormat object to the specified object.

Managing the NSOpenGLContext

`- prepareOpenGL` (page 1809)
Used by subclasses to initialize OpenGL state.

`- clearGLContext` (page 1807)
Releases the NSOpenGLContext object associated with the view.

`- openGLContext` (page 1808)
Returns the NSOpenGLContext object associated with the receiver.

`- setOpenGLContext:` (page 1810)
Sets the NSOpenGLContext object associated with the receiver.

Managing the Visible Region

`- reshape` (page 1810)
Called by Cocoa when the view's visible rectangle or bounds change.

`- update` (page 1811)
Called by Cocoa when the view's window moves or when the view itself moves or is resized.

Class Methods

defaultPixelFormat

Returns a default NSOpenGLPixelFormat object.

```
+ (NSOpenGLPixelFormat *)defaultPixelFormat
```

Return Value
A pixel format object with no attributes set.

Discussion
Typically used with the initializer `initWithFrame:pixelFormat:` (page 1808), this object has no attributes set.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `pixelFormat` (page 1809)
- `setPixelFormat:` (page 1811)

Related Sample Code
FunHouse
LiveVideoMixer
LiveVideoMixer2
LiveVideoMixer3
WhackedTV

Declared In
`NSOpenGLView.h`

Instance Methods

clearGLContext

Releases the NSOpenGLContext object associated with the view.

```
- (void)clearGLContext
```

Discussion
If necessary, this method calls the `clearDrawable` (page 1765) method of the context object before releasing it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `openGLContext` (page 1808)

- setOpenGLContext: (page 1810)

Declared In
NSOpenGLView.h

initWithFrame:pixelFormat:

Returns an NSOpenGLView object initialized with the specified frame rectangle and pixel format.

```
- (id)initWithFrame:(NSRect)frameRect pixelFormat:(NSOpenGLPixelFormat *)format
```

Parameters

frameRect
The frame rectangle for the view, specified in the coordinate system of its parent view.

format
The pixel format to use when creating the view's NSOpenGLContext object.

Return Value
An initialized NSOpenGLView object, or nil if the object could not be initialized.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ defaultPixelFormat (page 1807)

Related Sample Code
Cocoa OpenGL
GLSLShowpiece
LiveVideoMixer2
OpenCL NBody Simulation Example
VBL

Declared In
NSOpenGLView.h

openGLContext

Returns the NSOpenGLContext object associated with the receiver.

```
- (NSOpenGLContext *)openGLContext
```

Return Value
The OpenGL context object of the receiver.

Discussion
If the receiver has no associated context object, a new NSOpenGLContext object is created and returned. The new object is initialized with the receiver's pixel format information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `clearGLContext` (page 1807)
- `setOpenGLContext:` (page 1810)
- `pixelFormat` (page 1809)

Related Sample Code
Cocoa OpenGL
LiveVideoMixer
LiveVideoMixer2
NURBSSurfaceVertexProg
SurfaceVertexProgram

Declared In
`NSOpenGLView.h`

pixelFormat

Returns the NSOpenGLPixelFormat object associated with the receiver.

```
- (NSOpenGLPixelFormat *)pixelFormat
```

Return Value
The receiver's pixel format object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `+ defaultPixelFormat` (page 1807)
- `- initWithFrame:pixelFormat:` (page 1808)
- `- setPixelFormat:` (page 1811)

Related Sample Code
CIColorTracking
Cocoa OpenGL
CoreImageGLTextureFBO
DispatchFractal
Draw Pixels

Declared In
`NSOpenGLView.h`

prepareOpenGL

Used by subclasses to initialize OpenGL state.

```
- (void)prepareOpenGL
```

Discussion
This method is called only once after the OpenGL context is made the current context. Subclasses that implement this method can use it to configure the Open GL state in preparation for drawing.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
CoreImageGLTextureFBO
OpenCL NBody Simulation Example
QTCoreImage101
QTCoreVideo103
WhackedTV

Declared In
`NSOpenGLView.h`

reshape

Called by Cocoa when the view's visible rectangle or bounds change.

```
- (void)reshape
```

Discussion
Cocoa typically calls this method during scrolling and resize operations but may call it in other situations when the view's rectangles change. The default implementation does nothing. You can override this method if you need to adjust the viewport and display frustum.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DispatchLife
iChatTheater
NSOpenGL Fullscreen
VertexPerformanceDemo
VertexPerformanceTest

Declared In
`NSOpenGLView.h`

setOpenGLContext:

Sets the NSOpenGLContext object associated with the receiver.

```
- (void)setOpenGLContext:(NSOpenGLContext *)context
```

Parameters
context
The OpenGL context object to associate with the receiver.

Discussion
This method releases the current OpenGL context, if one already exists. You must also call the `setView:` (page 1775) method of the context object to synchronize the context with the view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `clearGLContext` (page 1807)
- `openGLContext` (page 1808)

Related Sample Code
LiveVideoMixer
LiveVideoMixer2
LiveVideoMixer3

Declared In
`NSOpenGLView.h`

setPixelFormat:

Sets the receiver's NSOpenGLPixelFormat object to the specified object.

```
- (void)setPixelFormat:(NSOpenGLPixelFormat *)pixelFormat
```

Parameters

pixelFormat
The new pixel format object for the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ `defaultPixelFormat` (page 1807)
- `pixelFormat` (page 1809)

Declared In
`NSOpenGLView.h`

update

Called by Cocoa when the view's window moves or when the view itself moves or is resized.

```
- (void)update
```

Discussion
The default implementation simply calls the `update` (page 1776) method of NSOpenGLContext. You can override this method to perform additional update operations on the context or if you need to add locks for multithreaded access to multiple contexts.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DispatchLife
LiveVideoMixer

LiveVideoMixer2
LiveVideoMixer3
OpenGL Filter Basics Cocoa

Declared In
`NSOpenGLView.h`

NSOpenPanel Class Reference

Inherits from	NSSavePanel : NSPanel : NSWindow : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSWindow) NSAnimatablePropertyContainer (NSWindow) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSOpenPanel.h
Companion guides	Application File Management Sheet Programming Topics
Related sample code	ExtractMovieAudioToAIFF FunHouse ImageKitDemo LSMSmartCategorizer QTExtractAndConvertToAIFF

Overview

The `NSOpenPanel` class provides the Open panel for the Cocoa user interface. Applications use the Open panel as a convenient way to query the user for the name of a file to open.

Tasks

Creating Panels

`+ openPanel` (page 1815)
Creates and returns a `NSOpenPanel` (page 1813) object.

Configuring Panels

- `canChooseFiles` (page 1818)
  Returns whether the panel allows the user to choose files to open.
- `setCanChooseFiles:` (page 1822)
  Sets whether the user can select files in the panel's browser.
- `canChooseDirectories` (page 1818)
  Returns whether the panel allows the user to choose directories to open.
- `setCanChooseDirectories:` (page 1822)
  Sets whether the user can select directories in the panel's browser.
- `resolvesAliases` (page 1819)
  Returns whether the panel resolves aliases.
- `setResolvesAliases:` (page 1823)
  Sets whether the panel resolves aliases.
- `allowsMultipleSelection` (page 1815)
  Returns whether the panel's browser allows the user to open multiple files (and directories) at a time.
- `setAllowsMultipleSelection:` (page 1821)
  Sets whether the user can select multiple files (and directories) at one time for opening.

Running Panels

- `beginForDirectory:file:types:modelessDelegate:didEndSelector:contextInfo:` (page 1816) Deprecated in Mac OS X v10.6
  Presents a modeless Open panel. (Deprecated. Use `beginWithCompletionHandler:` (page 2288) instead. You can set *`absoluteDirectoryPath`* using `setDirectoryURL:` (page 2300), and you can set *`fileTypes`* using `setAllowedFileTypes:` (page 2297).)
- `beginSheetForDirectory:file:types:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 1817) Deprecated in Mac OS X v10.6
  Presents an Open panel as a sheet with the directory specified by *`absoluteDirectoryPath`* and optionally the file specified by *`filename`* selected. (Deprecated. Use `beginSheetModalForWindow:completionHandler:` (page 2288) instead. You can set *`absoluteDirectoryPath`* using `setDirectoryURL:` (page 2300), and you can set *`fileTypes`* using `setAllowedFileTypes:` (page 2297).)
- `runModalForDirectory:file:types:` (page 1820) Deprecated in Mac OS X v10.6
  Displays the panel and begins a modal event loop that is terminated when the user clicks either OK or Cancel. (Deprecated. Use `runModal` (page 2295) instead. You can set *`path`* using `setDirectoryURL:` (page 2300), and you can set *`fileTypes`* using `setAllowedFileTypes:` (page 2297).)
- `runModalForTypes:` (page 1821) Deprecated in Mac OS X v10.6
  Displays the panel and begins a modal event loop that is terminated when the user clicks either OK or Cancel. (Deprecated. Use `runModal` (page 2295) instead. You can set *`fileTypes`* using `setAllowedFileTypes:` (page 2297).)

Accessing User Selection

- `URLs` (page 1823)
  Returns an array containing the absolute paths of the selected files and directories as URLs.
- `filenames` (page 1819) Deprecated in Mac OS X v10.6
  Returns an array containing the absolute paths (as NSString objects) of the selected files and directories. (Deprecated. Use `URLs` (page 1823) instead.)

Class Methods

openPanel

Creates and returns a `NSOpenPanel` (page 1813) object.

```
+ (NSOpenPanel *)openPanel
```

Return Value
The initialized Open panel.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
FunHouse
ImageKitDemo
LSMSmartCategorizer
QTKitPlayer
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSOpenPanel.h`

Instance Methods

allowsMultipleSelection

Returns whether the panel's browser allows the user to open multiple files (and directories) at a time.

```
- (BOOL)allowsMultipleSelection
```

Return Value
`YES` if the panel's browser allows multiple selection; otherwise, `NO`.

Discussion
If multiple files or directories are allowed, then the `filename` (page 2291) method—inherited from NSSavePanel—returns a non-`nil` value only if one and only one file is selected. By contrast, NSOpenPanel's `URLs` (page 1823) method always returns the URLs of the selected files, even if only one file is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `URL` (page 2307) (`NSSavePanel`)
- `URLs` (page 1823)
- `setAllowsMultipleSelection:` (page 1821)

Declared In
`NSOpenPanel.h`

beginForDirectory:file:types:modelessDelegate:didEndSelector:contextInfo:

Presents a modeless Open panel. (Deprecated in Mac OS X v10.6. Use `beginWithCompletionHandler:` (page 2288) instead. You can set *`absoluteDirectoryPath`* using `setDirectoryURL:` (page 2300), and you can set *`fileTypes`* using `setAllowedFileTypes:` (page 2297).)

```
- (void)beginForDirectory:(NSString *)absoluteDirectoryPath file:(NSString *)filename
     types:(NSArray *)fileTypes modelessDelegate:(id)modelessDelegate
    didEndSelector:(SEL)didEndSelector contextInfo:(void *)contextInfo
```

Parameters

`absoluteDirectoryPath`
The directory whose files the panel displays. When `nil`, the directory is the same directory used in the previous invocation of the panel; this is probably the best choice for most situations.

`filename`
Specifies a particular file in *`absoluteDirectoryPath`* that is selected when the Open panel is presented to the user. When `nil`, no file is initially selected.

`fileTypes`
An array of file extensions and/or HFS file types. Specifies the files the panel allows the user to select. `nil` makes all files in *`absoluteDirectoryPath`* selectable by the user. An array of types passed in here will override one set using `setAllowedFileTypes:` (page 2297).

`modelessDelegate`
This is not the same as a delegate assigned to the panel. This delegate is temporary and the relationship only lasts until the panel is dismissed.

`didEndSelector`
The message sent to *`modelessDelegate`* after the panel's session has ended, but before dismissing the Open panel. *`didEndSelector`* may dismiss the Open panel itself; otherwise, it will be dismissed on return from the method. The corresponding method should have the following signature:

```
- (void)openPanelDidEnd:(NSOpenPanel *)panel returnCode:(int)returnCode
contextInfo:(void  *)contextInfo
```

The value passed as *`returnCode`* will be either `NSCancelButton` or `NSOKButton`.

`contextInfo`
Any context information passed to *`modelessDelegate`* in the *`didEndSelector`* message.

Discussion
Similar to `beginSheetForDirectory:file:types:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 1817), but allows for modeless operation of the panel.

Availability
Available in Mac OS X v10.3 and later.
Deprecated in Mac OS X v10.6.

See Also
- `beginWithCompletionHandler:` (page 2288)

Declared In
`NSOpenPanel.h`

beginSheetForDirectory:file:types:modalForWindow:modalDelegate:didEndSelector:contextInfo:

Presents an Open panel as a sheet with the directory specified by *absoluteDirectoryPath* and optionally the file specified by *filename* selected. (Deprecated in Mac OS X v10.6. Use `beginSheetModalForWindow:completionHandler:` (page 2288) instead. You can set *absoluteDirectoryPath* using `setDirectoryURL:` (page 2300), and you can set *fileTypes* using `setAllowedFileTypes:` (page 2297).)

```
- (void)beginSheetForDirectory:(NSString *)absoluteDirectoryPath file:(NSString
    *)filename types:(NSArray *)fileTypes modalForWindow:(NSWindow *)docWindow
    modalDelegate:(id)modalDelegate didEndSelector:(SEL)didEndSelector
    contextInfo:(void *)contextInfo
```

Parameters

absoluteDirectoryPath
The directory whose files the panel displays. When `nil`, the directory is the same directory used in the previous invocation of the panel; this is probably the best choice for most situations.

filename
Specifies a particular file in *absoluteDirectoryPath* that is selected when the Open panel is presented to the user. When `nil`, no file is initially selected.

fileTypes
An array of file extensions and/or HFS file types. Specifies the files the panel allows the user to select. nil makes all files in *absoluteDirectoryPath* selectable by the user. An array of types passed in here will override one set using `setAllowedFileTypes:` (page 2297).

docWindow
The window to open the sheet on.

modalDelegate
This is not the same as a delegate assigned to the panel. This delegate is temporary and the relationship only lasts until the panel is dismissed..

didEndSelector
The message sent to *modalDelegate* after the modal session has ended, but before dismissing the Open panel. *didEndSelector* may dismiss the Open panel itself; otherwise, it will be dismissed on return from the method. The corresponding method should have the following signature:

```
- (void)openPanelDidEnd:(NSOpenPanel *)panel returnCode:(int)returnCode
contextInfo:(void  *)contextInfo
```

The value passed as *returnCode* will be either `NSCancelButton` or `NSOKButton`.

contextInfo
Any context information passed to *modalDelegate* in the *didEndSelector* message.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- beginSheetModalForWindow:completionHandler: (page 2288)

Related Sample Code
DemoAssistant
IKImageViewDemo
QTExtractAndConvertToMovieFile
SillyFrequencyLevels
XMLBrowser

Declared In
NSOpenPanel.h

canChooseDirectories

Returns whether the panel allows the user to choose directories to open.

```
- (BOOL)canChooseDirectories
```

Return Value
YES if the panel allows the user to choose directories; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setCanChooseDirectories: (page 1822)

Declared In
NSOpenPanel.h

canChooseFiles

Returns whether the panel allows the user to choose files to open.

```
- (BOOL)canChooseFiles
```

Return Value
YES if the panel allows the user to choose files; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setCanChooseFiles: (page 1822)

Declared In
NSOpenPanel.h

filenames

Returns an array containing the absolute paths (as NSString objects) of the selected files and directories. (Deprecated in Mac OS X v10.6. Use URLs (page 1823) instead.)

```
- (NSArray *)filenames
```

Return Value
The array of filenames.

Discussion
If multiple selections aren't allowed, the array contains a single name. The filenames method is preferable over NSSavePanel's filename (page 2291) to get the name or names of files and directories that the user has selected.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- URLs (page 1823)

Related Sample Code
FunHouse
ImageKitDemo
LSMSmartCategorizer
ThreadsImporter
ThreadsImportMovie

Declared In
NSOpenPanel.h

resolvesAliases

Returns whether the panel resolves aliases.

```
- (BOOL)resolvesAliases
```

Return Value
YES if the panel resolves aliases; otherwise, NO.

Discussion
If YES, the effect is that dropping an alias on the panel or asking for filenames or URLs returns the resolved aliases. The default is YES.

Availability
Available in Mac OS X v10.1 and later.

See Also
- setResolvesAliases: (page 1823)

Declared In
NSOpenPanel.h

runModalForDirectory:file:types:

Displays the panel and begins a modal event loop that is terminated when the user clicks either OK or Cancel. (Deprecated in Mac OS X v10.6. Use runModal (page 2295) instead. You can set *path* using setDirectoryURL: (page 2300), and you can set *fileTypes* using setAllowedFileTypes: (page 2297).)

```
- (NSInteger)runModalForDirectory:(NSString *)absoluteDirectoryPath file:(NSString
    *)filename types:(NSArray *)fileTypes
```

Parameters

absoluteDirectoryPath
The directory whose files the panel displays. When nil, the directory is the same directory used in the previous invocation of the panel; this is probably the best choice for most situations.

filename
Specifies a particular file in *absoluteDirectoryPath* that is selected when the Open panel is presented to the user. When nil, no file is initially selected.

fileTypes
An array of file extensions and/or HFS file types. Specifies the files the panel allows the user to select. nil makes all files in *absoluteDirectoryPath* selectable by the user. An array of types passed in here will override one set using setAllowedFileTypes: (page 2297).

Return Value
The button clicked to dismiss the dialog: NSOKButton for the OK button and NSCancelButton for the Cancel button.

Discussion
You can control whether directories and files appear in the browser with the setCanChooseDirectories: (page 1822) and setCanChooseFiles: (page 1822) methods.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- runModal (page 2295)

Related Sample Code
CocoaVideoFrameToGWorld
LiveVideoMixer
LiveVideoMixer2
QTGraphicsImport
SBSetFinderComment

Declared In
NSOpenPanel.h

runModalForTypes:

Displays the panel and begins a modal event loop that is terminated when the user clicks either OK or Cancel. (Deprecated in Mac OS X v10.6. Use `runModal` (page 2295) instead. You can set *fileTypes* using `setAllowedFileTypes:` (page 2297).)

```
- (NSInteger)runModalForTypes:(NSArray *)fileTypes
```

Parameters

fileTypes

An array of file extensions and/or HFS file types. Specifies the files the panel allows the user to select. `nil` makes all files selectable by the user. An array of types passed in here will override one set using `setAllowedFileTypes:` (page 2297).

Return Value

The button used to dismiss the dialog: `NSOKButton` for the OK button and `NSCancelButton` for the Cancel button.

Discussion

This convenience method sends `runModalForDirectory:nil file:nil types:`*fileTypes* to the panel. See `runModalForDirectory:file:types:` (page 1820) for additional details.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- `runModal` (page 2295)

Related Sample Code

FunHouse

ImageKitDemo

SBSendEmail

ThreadsExportMovie

ThreadsImportMovie

Declared In

`NSOpenPanel.h`

setAllowsMultipleSelection:

Sets whether the user can select multiple files (and directories) at one time for opening.

```
- (void)setAllowsMultipleSelection:(BOOL)flag
```

Parameters

flag

If `YES`, the panel's browser allows multiple selection; if `NO`, it does not.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `allowsMultipleSelection` (page 1815)

Related Sample Code
AudioBurn
ContentBurn
FunHouse
ObjectPath
Quartz Composer WWDC 2005 TextEdit

Declared In
NSOpenPanel.h

setCanChooseDirectories:

Sets whether the user can select directories in the panel's browser.

```
- (void)setCanChooseDirectories:(BOOL)flag
```

Parameters

flag

If YES, the panel allows the user to choose directories; if NO, it does not.

Discussion
When a directory is selected, the OK button is enabled only if *flag* is YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- canChooseDirectories (page 1818)

Related Sample Code
ContentBurn
ImageKitDemo
LSMSmartCategorizer
MovieAssembler
ObjectPath

Declared In
NSOpenPanel.h

setCanChooseFiles:

Sets whether the user can select files in the panel's browser.

```
- (void)setCanChooseFiles:(BOOL)flag
```

Parameters

flag

If YES, the panel allows the user to choose files; if NO, it does not.

Availability
Available in Mac OS X v10.0 and later.

See Also
- canChooseFiles (page 1818)

Related Sample Code
ContentBurn
ImageKitDemo
LSMSmartCategorizer
MovieAssembler
ObjectPath

Declared In
NSOpenPanel.h

setResolvesAliases:

Sets whether the panel resolves aliases.

```
- (void)setResolvesAliases:(BOOL)resolvesAliases
```

Parameters
resolvesAliases
If YES, the panel resolves aliases; if NO, it does not.

Discussion
If YES, the effect is that dropping an alias on the panel or asking for filenames or URLs returns the resolved aliases. Set this value to NO to allow selection of aliases without resolving.

Availability
Available in Mac OS X v10.1 and later.

See Also
- resolvesAliases (page 1819)

Related Sample Code
CIFilterGeneratorTest
CIVideoDemoGL
DesktopImage
ObjectPath
Quartz 2D Transformer

Declared In
NSOpenPanel.h

URLs

Returns an array containing the absolute paths of the selected files and directories as URLs.

```
- (NSArray *)URLs
```

Return Value
The array of URLs.

Discussion
If multiple selections aren't allowed, the array contains a single name.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaSpeechSynthesisExample
IKImageViewDemo
QTKitFrameStepper
QTMetadataEditor
SBSetFinderComment

Declared In
`NSOpenPanel.h`

NSOutlineView Class Reference

Inherits from	NSTableView : NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSTableView) NSTextViewDelegate (NSTableView) NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSOutlineView.h
Companion guides	Outline View Programming Topics Drag and Drop Programming Topics for Cocoa
Related sample code	ColorSyncDevices-Cocoa DragNDropOutlineView EnhancedDataBurn QTKitMovieShuffler SGDevices

Overview

`NSOutlineView` is a subclass of `NSTableView` that uses a row-and-column format to display hierarchical data that can be expanded and collapsed, such as directories and files in a file system. A user can expand and collapse rows, edit values, and resize and rearrange columns.

Like a table view, an outline view does not store its own data, instead it retrieves data values as needed from a data source to which it has a weak reference (see Communicating With Objects). See the `NSOutlineViewDataSource` protocol, which declares the methods that an `NSOutlineView` object uses to access the contents of its data source object.

An outline view has the following features:

- A user can expand and collapse rows.
- Each item in the outline view must be unique. In order for the collapsed state to remain consistent between reloads the item's pointer must remain the same.
- The view gets data from a data source (see the `NSOutlineViewDataSource` protocol).
- The view retrieves only the data that needs to be displayed.

Important: It is possible that your datasource methods for populating the outline view may be called before `awakeFromNib` (page 3731) is called if the datasource is specified in Interface Builder. You should defend against this by having the datasource's `outlineView:numberOfChildrenOfItem:` (page 3746) method return `0` for the number of children when the datasource has not yet been configured. In `awakeFromNib` (page 3731), when the datasource is initialized you should always call `reloadData` (page 2645) (inherited from the superclass `NSTableView`) on the resulting outline view.

Delegation

The `NSOutlineView` provides a number of customization features through delegation. The majority of the delegate methods are declared in the `NSOutlineViewDelegate Protocol`. However, due to an error in the transition, some methods are still an informal protocol and appear in this document as delegate methods. All these methods are `NSNotificationCenter` oriented and are the following methods:

- `outlineViewColumnDidMove:` (page 1844)
- `outlineViewColumnDidResize:` (page 1844)
- `outlineViewItemDidCollapse:` (page 1844)
- `outlineViewItemDidExpand:` (page 1845)
- `outlineViewItemWillCollapse:` (page 1845)
- `outlineViewItemWillExpand:` (page 1845)
- `outlineViewSelectionDidChange:` (page 1846)
- `outlineViewSelectionIsChanging:` (page 1846)

Tasks

Setting the Data Source

- `setDataSource:` (page 1841)
  Sets the receiver's data source to a given object.
- `dataSource` (page 1832)
  Returns the object that provides the data displayed by the receiver.

Working with Expandability

- `isExpandable:` (page 1835)
  Returns a Boolean value that indicates whether a given item is expandable.
- `isItemExpanded:` (page 1835)
  Returns a Boolean value that indicates whether a given item is expanded.

- `outlineViewItemWillExpand:` (page 1845) *delegate method*
  Invoked when *notification* is posted—that is, whenever the user is about to expand an item in the outline view.
- `outlineViewItemDidExpand:` (page 1845) *delegate method*
  Invoked when *notification* is posted—that is, whenever the user expands an item in the outline view.
- `outlineViewItemWillCollapse:` (page 1845) *delegate method*
  Invoked when *notification* is posted—that is, whenever the user is about to collapse an item in the outline view.
- `outlineViewItemDidCollapse:` (page 1844) *delegate method*
  Invoked when *notification* is posted—that is, whenever the user collapses an item in the outline view.

Monitoring Selection Changes

- `outlineViewSelectionIsChanging:` (page 1846) *delegate method*
  Invoked when *notification* is posted—that is, whenever the outline view's selection changes.
- `outlineViewSelectionDidChange:` (page 1846) *delegate method*
  Invoked when *notification* is posted—that is, immediately after the outline view's selection has changed.

Expanding and Collapsing the Outline

- `expandItem:` (page 1832)
  Expands a given item.
- `expandItem:expandChildren:` (page 1833)
  Expands a specified item and, optionally, its children.
- `collapseItem:` (page 1830)
  Collapses a given item.
- `collapseItem:collapseChildren:` (page 1831)
  Collapses a given item and, optionally, its children.

Redisplaying Information

- `reloadItem:` (page 1838)
  Reloads and redisplays the data for *item*.
- `reloadItem:reloadChildren:` (page 1838)
  Reloads a given item and, optionally, its children.

Converting Between Items and Rows

- `itemAtRow:` (page 1835)
  Returns the item associated with a given row.

- `rowForItem:` (page 1839)
  Returns the row associated with a given item.

Working with the Outline Column

- `setOutlineTableColumn:` (page 1843)
  Sets the table column in which hierarchical data is displayed.
- `outlineTableColumn` (page 1837)
  Returns the table column in which hierarchical data is displayed.
- `autoresizesOutlineColumn` (page 1829)
  Returns a Boolean value that indicates whether the receiver automatically resizes its outline column when the user expands or collapses items.
- `setAutoresizesOutlineColumn:` (page 1839)
  Sets whether the receiver automatically resizes its outline column when the user expands or collapses an item.
- `outlineViewColumnDidMove:` (page 1844) *delegate method*
  Invoked when *`notification`* is posted—that is, whenever the user moves a column in the outline view.
- `outlineViewColumnDidResize:` (page 1844) *delegate method*
  Invoked when *`notification`* is posted—that is, whenever the user resizes a column in the outline view.

Working with Indentation

- `levelForItem:` (page 1836)
  Returns the indentation level for a given item.
- `levelForRow:` (page 1836)
  Returns the indentation level for a given row.
- `setIndentationPerLevel:` (page 1842)
  Sets the per-level indentation.
- `indentationPerLevel` (page 1834)
  Returns the current indentation per level.
- `setIndentationMarkerFollowsCell:` (page 1842)
  Sets whether the indentation marker symbol displayed in the outline column should be indented along with the cell contents, or always displayed left-justified in the column.
- `indentationMarkerFollowsCell` (page 1834)
  Returns a Boolean value that indicates whether the indentation marker symbol displayed in the outline column should be indented along with the cell contents, or always displayed left-justified in the column.

Working with Persistence

- `autosaveExpandedItems` (page 1830)
  Returns a Boolean value that indicates whether the expanded items in the receiver are automatically saved.
- `setAutosaveExpandedItems:` (page 1840)
  Sets whether the expanded items in the receiver are automatically saved.

Supporting Drag and Drop

- `setDropItem:dropChildIndex:` (page 1841)
  Used to “retarget” a proposed drop.
- `shouldCollapseAutoExpandedItemsForDeposited:` (page 1843)
  Returns a Boolean value that indicates whether auto-expanded items should return to their original collapsed state.

Getting the Parent for an Item

- `parentForItem:` (page 1837)
  Returns the parent for a given item.

Getting the Frame for a Cell

- `frameOfOutlineCellAtRow:` (page 1833)
  Returns the frame of the outline cell for a given row.

Getting and Setting the Delegate

- `delegate` (page 1832)
  Returns the receiver’s delegate.
- `setDelegate:` (page 1841)
  Sets the receiver’s delegate.

Instance Methods

autoresizesOutlineColumn

Returns a Boolean value that indicates whether the receiver automatically resizes its outline column when the user expands or collapses items.

```
- (BOOL)autoresizesOutlineColumn
```

Return Value
`YES` if the outline column is automatically resized, otherwise `NO`.

Discussion
The outline column contains the cells with the expansion symbols and is generally the first column. The default is `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAutoresizesOutlineColumn:` (page 1839)

Declared In
`NSOutlineView.h`

autosaveExpandedItems

Returns a Boolean value that indicates whether the expanded items in the receiver are automatically saved.

```
- (BOOL)autosaveExpandedItems
```

Return Value
`YES` if when an item is expanded, the outline view displays the previous expanded state of its contained items, otherwise `NO`.

Discussion
The outline view information is saved separately for each user and for each application that user uses. Note that if `autosaveName` (page 2622) returns `nil`, this setting is ignored, and outline information isn't saved.

Special Considerations

Starting in Mac OS X version 10.5, the value for `autosaveExpandedItems` is saved out in the nib file. The default value is `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- autosaveName` (page 2622) (`NSTableView`)
`- autosaveTableColumns` (page 2623) (`NSTableView`)
`- setAutosaveExpandedItems:` (page 1840)

Declared In
`NSOutlineView.h`

collapseItem:

Collapses a given item.

```
- (void)collapseItem:(id)item
```

Parameters

item

An item in the receiver.

Discussion

If *item* is not expanded or not expandable, does nothing

If collapsing takes place, posts item collapse notification.

Availability

Available in Mac OS X v10.0 and later.

See Also

- expandItem: (page 1832)

Related Sample Code

DragNDropOutlineView

Declared In

NSOutlineView.h

collapseItem:collapseChildren:

Collapses a given item and, optionally, its children.

```
- (void)collapseItem:(id)item collapseChildren:(BOOL)collapseChildren
```

Parameters

item

An item in the receiver.

Starting in Mac OS X version 10.5, passing 'nil' will collapse each item under the root in the outline view.

collapseChildren

If YES, recursively collapses *item* and its children. If NO, collapses *item* only (identical to collapseItem: (page 1830)).

Discussion

For example, this method is invoked with the *collapseChildren* parameter set to YES when a user Option-clicks the disclosure triangle for an item in the outline view (to collapse the item and all its contained items).

For each item collapsed, posts an item collapsed notification.

Availability

Available in Mac OS X v10.0 and later.

See Also

- collapseItem: (page 1830)
- expandItem:expandChildren: (page 1833)

Declared In

NSOutlineView.h

dataSource

Returns the object that provides the data displayed by the receiver.

```
- (id < NSOutlineViewDataSource >)dataSource
```

Return Value
The object that provides the data displayed by the receiver.

Discussion
See Writing an Outline View Data Source and the NSOutlineViewDataSource Protocol Reference (page 3741) informal protocol specification for more information.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setDataSource: (page 1841)

Declared In
NSOutlineView.h

delegate

Returns the receiver's delegate.

```
- (id < NSOutlineViewDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setDelegate: (page 1841)

Declared In
NSOutlineView.h

expandItem:

Expands a given item.

```
- (void)expandItem:(id)item
```

Parameters
item
An item in the receiver.

Discussion
If *item* is not expandable or is already expanded, does nothing.

If expanding takes place, posts an item expanded notification.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `collapseItem:` (page 1830)

Related Sample Code
DragNDropOutlineView

Declared In
`NSOutlineView.h`

expandItem:expandChildren:

Expands a specified item and, optionally, its children.

```
- (void)expandItem:(id)item expandChildren:(BOOL)expandChildren
```

Parameters

item

An item in the receiver.

Starting in Mac OS X version 10.5, passing `'nil'` will expand each item under the root in the outline view.

expandChildren

If `YES`, recursively expands *item* and its children. If `NO`, expands *item* only (identical to `expandItem:` (page 1832)).

Discussion
For example, this method is invoked with the *expandChildren* parameter set to `YES` when a user Option-clicks the disclosure triangle for an item in the outline view (to expand the item and all its contained items).

For each item expanded, posts an item expanded notification.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `collapseItem:collapseChildren:` (page 1831)
- `expandItem:` (page 1832)

Declared In
`NSOutlineView.h`

frameOfOutlineCellAtRow:

Returns the frame of the outline cell for a given row.

```
- (NSRect)frameOfOutlineCellAtRow:(NSInteger)row
```

Parameters

row

The index of the row for which to return the frame.

Return Value
The frame of the outline cell for the row at index *row*, considering the current indentation and the value returned by `indentationMarkerFollowsCell` (page 1834). If the row at index *row* is not an expandable row, returns `NSZeroRect`.

Discussion
You can override this method in a subclass to return a custom frame for the outline button cell. If your override returns an empty rect, no outline cell is drawn for that row. You might do that, for example, so that the disclosure triangle will not be shown for a row that should never be expanded.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSOutlineView.h`

indentationMarkerFollowsCell

Returns a Boolean value that indicates whether the indentation marker symbol displayed in the outline column should be indented along with the cell contents, or always displayed left-justified in the column.

```
- (BOOL)indentationMarkerFollowsCell
```

Return Value
`YES` if the indentation marker is indented along with the cell contents, otherwise `NO`.

Discussion
The default is `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setIndentationMarkerFollowsCell:` (page 1842)

Declared In
`NSOutlineView.h`

indentationPerLevel

Returns the current indentation per level.

```
- (CGFloat)indentationPerLevel
```

Return Value
The current indentation per level, in points.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setIndentationPerLevel:` (page 1842)

Declared In
NSOutlineView.h

isExpandable:

Returns a Boolean value that indicates whether a given item is expandable.

```
- (BOOL)isExpandable:(id)item
```

Parameters

item
An item in the receiver.

Return Value
YES if *item* is expandable—that is, *item* can contain other items, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- expandItem: (page 1832)
- isItemExpanded: (page 1835)

Declared In
NSOutlineView.h

isItemExpanded:

Returns a Boolean value that indicates whether a given item is expanded.

```
- (BOOL)isItemExpanded:(id)item
```

Parameters

item
An item in the receiver.

Return Value
YES if *item* is expanded, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- expandItem: (page 1832)
- isExpandable: (page 1835)

Declared In
NSOutlineView.h

itemAtRow:

Returns the item associated with a given row.

```
- (id)itemAtRow:(NSInteger)row
```

Parameters

row

The index of a row in the receiver.

Return Value

The item associated with *row*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- rowForItem: (page 1839)

Related Sample Code

DragNDropOutlineView

PhotoSearch

SourceView

Declared In

NSOutlineView.h

levelForItem:

Returns the indentation level for a given item.

```
- (NSInteger)levelForItem:(id)item
```

Parameters

item

An item in the receiver.

Return Value

The indentation level for *item*. If *item* is nil (which is the root item), returns -1.

Discussion

The levels are zero-based—that is, the first level of displayed items is level 0.

Availability

Available in Mac OS X v10.0 and later.

See Also

- indentationPerLevel (page 1834)
- levelForRow: (page 1836)

Declared In

NSOutlineView.h

levelForRow:

Returns the indentation level for a given row.

```
- (NSInteger)levelForRow:(NSInteger)row
```

Parameters

row

The index of a row in the receiver.

Return Value

The indentation level for *row*. For an invalid row, returns `-1`.

Discussion

The levels are zero-based—that is, the first level of displayed items is level `0`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- indentationPerLevel` (page 1834)

`- levelForItem:` (page 1836)

Declared In

`NSOutlineView.h`

outlineTableColumn

Returns the table column in which hierarchical data is displayed.

```
- (NSTableColumn *)outlineTableColumn
```

Return Value

The table column in which hierarchical data is displayed.

Discussion

Each level of hierarchical data is indented by the amount specified by `indentationPerLevel` (page 1834) (the default is 16.0), and decorated with the indentation marker (disclosure triangle) on rows that are expandable.

Special Considerations

Starting in Mac OS X version 10.5, outline table column data is saved in `encodeWithCoder:` and restored in `initWithCoder:`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setOutlineTableColumn:` (page 1843)

Declared In

`NSOutlineView.h`

parentForItem:

Returns the parent for a given item.

```
- (id)parentForItem:(id)item
```

Parameters

item

The item for which to return the parent.

Return Value

The parent for *item*, or `nil` if the parent is the root.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSOutlineView.h`

reloadItem:

Reloads and redisplays the data for *item*.

```
- (void)reloadItem:(id)item
```

Parameters

item

The item to reload and display

Discussion

This method may cause the outline view to change its selection unexpectedly.

Special Considerations

In Mac OS X v 10.6 and earlier, this method will not invoke the `outlineViewSelectionDidChange:` (page 1846) delegate method.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- reloadItem:reloadChildren:` (page 1838)

Related Sample Code

NewsReader

Declared In

`NSOutlineView.h`

reloadItem:reloadChildren:

Reloads a given item and, optionally, its children.

```
- (void)reloadItem:(id)item reloadChildren:(BOOL)reloadChildren
```

Parameters

item

An item in the receiver.

Starting in Mac OS X version 10.5, passing `'nil'` will reload everything under the root in the outline view.

reloadChildren

If `YES`, recursively reloads *item* and its children. If `NO`, reloads *item* only (identical to `reloadItem:` (page 1838)).

It is not necessary, or efficient, to reload children if the item is not expanded.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- reloadItem:` (page 1838)

Related Sample Code
PhotoSearch

Declared In
`NSOutlineView.h`

rowForItem:

Returns the row associated with a given item.

```
- (NSInteger)rowForItem:(id)item
```

Parameters

item

An item in the receiver.

Return Value
The row associated with *item*, or `-1` if *item* is `nil` or cannot be found.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- itemAtRow:` (page 1835)

Related Sample Code
DragNDropOutlineView

Declared In
`NSOutlineView.h`

setAutoresizesOutlineColumn:

Sets whether the receiver automatically resizes its outline column when the user expands or collapses an item.

```
- (void)setAutoresizesOutlineColumn:(BOOL)resize
```

Parameters

resize

YES if the outline column is automatically resized, otherwise NO.

Discussion

The outline column contains the cells with the expansion symbols and is generally the first column. The default is YES.

Availability

Available in Mac OS X v10.0 and later.

See Also

- autoresizesOutlineColumn (page 1829)

Declared In

NSOutlineView.h

setAutosaveExpandedItems:

Sets whether the expanded items in the receiver are automatically saved.

```
- (void)setAutosaveExpandedItems:(BOOL)flag
```

Discussion

If *flag* is different from the current value, this method also reads in the saved information and sets the outline view's options to match. YES indicates that when an item is expanded, the outline view displays the previous expanded state of its contained items.

The outline information is saved separately for each user and for each application that user uses.

If autosaveName (page 2622) returns nil or if you haven't implemented the data source methods outlineView:itemForPersistentObject: and outlineView:persistentObjectForItem:, this setting is ignored, and expanded item information isn't saved.

Note that you can have separate settings for autosaveExpandedItems (page 1830) and autosaveTableColumns (page 2623), so you could, for example, save expanded item information, but not table column positions.

Special Considerations

Starting in Mac OS X version 10.5, the value for autosaveExpandedItems is saved out in the nib file. The default value is NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- autosaveExpandedItems (page 1830)
- setAutosaveTableColumns: (page 2658) (NSTableView)

Declared In

NSOutlineView.h

setDataSource:

Sets the receiver's data source to a given object.

```
- (void)setDataSource:(id < NSOutlineViewDataSource >)anObject
```

Parameters

anObject

The data source for the receiver. The object must implement the appropriate methods of the NSOutlineViewDataSource Protocol Reference (page 3741) informal protocol.

Discussion

The receiver maintains a weak reference to the data source (see Communicating With Objects). After setting the data source, this method invokes `tile` (page 2673).

This method raises an `NSInternalInconsistencyException` if *anObject* doesn't respond to all of `outlineView:child:ofItem:`, `outlineView:isItemExpandable:`, `outlineView:numberOfChildrenOfItem:`, and `outlineView:objectValueForTableColumn:byItem:`.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- dataSource` (page 1832)

Declared In

`NSOutlineView.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSOutlineViewDelegate >)anObject
```

Parameters

anObject

The delegate for the receiver. The delegate must conform to the `NSOutlineViewDelegate Protocol` protocol.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- delegate` (page 1832)

Declared In

`NSOutlineView.h`

setDropItem:dropChildIndex:

Used to "retarget" a proposed drop.

```
- (void)setDropItem:(id)item dropChildIndex:(NSInteger)index
```

Parameters

item

The target item.

index

The drop index.

Discussion

For example, to specify a drop on `someOutlineItem`, you specify *item* as `someOutlineItem` and *index* as `NSOutlineViewDropOnItemIndex`. To specify a drop between child `2` and `3` of `someOutlineItem`, you specify *item* as `someOutlineItem` and *index* as `3` (children are a zero-based index). To specify a drop on an item that can't be expanded `someOutlineItem`, you specify *item* as `someOutlineItem` and *index* as `NSOutlineViewDropOnItemIndex`.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

DragNDropOutlineView

Declared In

`NSOutlineView.h`

setIndentationMarkerFollowsCell:

Sets whether the indentation marker symbol displayed in the outline column should be indented along with the cell contents, or always displayed left-justified in the column.

```
- (void)setIndentationMarkerFollowsCell:(BOOL)drawInCell
```

Discussion

The default is `YES`, the indentation marker is indented along with the cell contents.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- indentationMarkerFollowsCell` (page 1834)

Declared In

`NSOutlineView.h`

setIndentationPerLevel:

Sets the per-level indentation.

```
- (void)setIndentationPerLevel:(CGFloat)newIndentLevel
```

Parameters

newIndentLevel

The indentation per level, in points.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `indentationPerLevel` (page 1834)

Declared In
`NSOutlineView.h`

setOutlineTableColumn:

Sets the table column in which hierarchical data is displayed.

```
- (void)setOutlineTableColumn:(NSTableColumn *)outlineTableColumn
```

Parameters

outlineTableColumn
The table column in which hierarchical data is displayed.

Special Considerations

Starting in Mac OS X version 10.5, outline table column data is saved in `encodeWithCoder:` and restored in `initWithCoder:`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `outlineTableColumn` (page 1837)

Declared In
`NSOutlineView.h`

shouldCollapseAutoExpandedItemsForDeposited:

Returns a Boolean value that indicates whether auto-expanded items should return to their original collapsed state.

```
- (BOOL)shouldCollapseAutoExpandedItemsForDeposited:(BOOL)deposited
```

Return Value
`YES` if auto expanded items should return to their original collapsed state, otherwise `NO`.

Discussion
Override this method to provide custom behavior. *deposited* tells whether or not the drop terminated due to a successful drop.

This method is called in a variety of situations. For example, it is sent shortly after `outlineView:acceptDrop:item:childIndex:` is processed and also if the drag exits the outline view (exiting the view is treated the same as a failed drop).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

Delegate Methods

outlineViewColumnDidMove:

Invoked when *notification* is posted—that is, whenever the user moves a column in the outline view.

```
- (void)outlineViewColumnDidMove:(NSNotification *)notification
```

Discussion
This method is invoked as a result of posting an `NSOutlineViewColumnDidMoveNotification` (page 1847).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

outlineViewColumnDidResize:

Invoked when *notification* is posted—that is, whenever the user resizes a column in the outline view.

```
- (void)outlineViewColumnDidResize:(NSNotification *)notification
```

Discussion
This method is invoked as a result of posting an `NSOutlineViewColumnDidResizeNotification` (page 1847).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

outlineViewItemDidCollapse:

Invoked when *notification* is posted—that is, whenever the user collapses an item in the outline view.

```
- (void)outlineViewItemDidCollapse:(NSNotification *)notification
```

Discussion
This method is invoked as a result of posting an `NSOutlineViewItemDidCollapseNotification` (page 1847).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

outlineViewItemDidExpand:

Invoked when *notification* is posted—that is, whenever the user expands an item in the outline view.

```
- (void)outlineViewItemDidExpand:(NSNotification *)notification
```

Discussion
This method is invoked as a result of posting an `NSOutlineViewItemDidExpandNotification` (page 1848).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

outlineViewItemWillCollapse:

Invoked when *notification* is posted—that is, whenever the user is about to collapse an item in the outline view.

```
- (void)outlineViewItemWillCollapse:(NSNotification *)notification
```

Discussion
This method is invoked as a result of posting an `NSOutlineViewItemWillCollapseNotification` (page 1848).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

outlineViewItemWillExpand:

Invoked when *notification* is posted—that is, whenever the user is about to expand an item in the outline view.

```
- (void)outlineViewItemWillExpand:(NSNotification *)notification
```

Discussion
This method is invoked as a result of posting an `NSOutlineViewItemWillExpandNotification` (page 1848).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

outlineViewSelectionDidChange:

Invoked when *notification* is posted—that is, immediately after the outline view's selection has changed.

```
- (void)outlineViewSelectionDidChange:(NSNotification *)notification
```

Discussion
This method is invoked as a result of posting an `NSOutlineViewSelectionDidChangeNotification` (page 1849).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

outlineViewSelectionIsChanging:

Invoked when *notification* is posted—that is, whenever the outline view's selection changes.

```
- (void)outlineViewSelectionIsChanging:(NSNotification *)notification
```

Discussion
This method is invoked as a result of posting an `NSOutlineViewSelectionIsChangingNotification` (page 1849).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

Constants

Drop on Item Index

This constant defines an index that allows you to drop an item directly on a target.

```
enum {
   NSOutlineViewDropOnItemIndex = -1
};
```

Constants
`NSOutlineViewDropOnItemIndex`
May be used as a valid child index of a drop target item.
In this case, the drop will happen directly on the target item.
Available in Mac OS X v10.0 and later.
Declared in `NSOutlineView.h`.

Declared In
`NSOutlineView.h`

Notifications

NSOutlineViewColumnDidMoveNotification

Posted whenever a column is moved by user action in an `NSOutlineView` object.

The notification object is the `NSOutlineView` object in which a column moved. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSOldColumn"`	An `NSNumber` object containing the integer value of the column's original index
`@"NSNewColumn"`	An `NSNumber` object containing the integer value of the column's present index

Availability
Available in Mac OS X v10.0 and later.

See Also
`- moveColumn:toColumn:` (page 2640) (`NSTableView`)

Declared In
`NSOutlineView.h`

NSOutlineViewColumnDidResizeNotification

Posted whenever a column is resized in an `NSOutlineView` object.

The notification object is the `NSOutlineView` object in which a column was resized. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSTableColumn"`	The column that was resized.
`@"NSOldWidth"`	An `NSNumber` object containing the column's original width

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

NSOutlineViewItemDidCollapseNotification

Posted whenever an item is collapsed in an `NSOutlineView` object.

The notification object is the `NSOutlineView` object in which an item was collapsed. A collapsed item's children lose their status as being selected. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSObject"`	The item that was collapsed (an `id`)

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

NSOutlineViewItemDidExpandNotification

Posted whenever an item is expanded in an `NSOutlineView` object.

The notification object is the `NSOutlineView` object in which an item was expanded. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSObject"`	The item that was expanded (an `id`)

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

NSOutlineViewItemWillCollapseNotification

Posted before an item is collapsed (after the user clicks the arrow but before the item is collapsed).

The notification object is the `NSOutlineView` object that contains the item about to be collapsed. A collapsed item's children will lose their status as being selected. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSObject"`	The item about to be collapsed (an `id`)

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOutlineView.h`

NSOutlineViewItemWillExpandNotification

Posted before an item is expanded (after the user clicks the arrow but before the item is collapsed).

The notification object is the outline view that contains an item about to be expanded. The *userInfo* dictionary contains the following information:

Key	Value
@"NSObject"	The item that is to be expanded (an id)

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSOutlineView.h

NSOutlineViewSelectionDidChangeNotification

Posted after the outline view's selection changes.

The notification object is the outline view whose selection changed. This notification does not contain a *userInfo* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSOutlineView.h

NSOutlineViewSelectionIsChangingNotification

Posted as the outline view's selection changes (while the mouse button is still down).

The notification object is the outline view whose selection is changing. This notification does not contain a *userInfo* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSOutlineView.h

NSPageLayout Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSPageLayout.h
Companion guide	Printing Programming Topics for Cocoa
Related sample code	From A View to A Movie From A View to A Picture GLSL Showpiece Lite GLUT Quartz Composer WWDC 2005 TextEdit

Overview

`NSPageLayout` is a panel that queries the user for information such as paper type and orientation. It is normally displayed in response to the user selecting the Page Setup menu item. You obtain an instance with the `pageLayout` (page 1853) class method. The pane can then be run as a sheet using `beginSheetWithPrintInfo:modalForWindow:delegate:didEndSelector:contextInfo:` (page 1854) or modally using `runModal` (page 1856) or `runModalWithPrintInfo:` (page 1857).

Tasks

Creating an NSPageLayout Instance

`+ pageLayout` (page 1853)
Returns a newly created `NSPageLayout` object.

Running a Page Setup Dialog

- `beginSheetWithPrintInfo:modalForWindow:delegate:didEndSelector:contextInfo:` (page 1854)
  Presents a page setup sheet for the given `NSPrintInfo` object, document-modal relative to the given window.
- `runModal` (page 1856)
  Displays the receiver and begins the modal loop using the shared `NSPrintInfo` object.
- `runModalWithPrintInfo:` (page 1857)
  Displays the receiver and begins the modal loop using the given `NSPrintInfo` object.

Customizing the Page Setup Dialog

- `addAccessoryController:` (page 1854)
  Adds the given controller of an accessory view to be presented in the page setup panel.
- `removeAccessoryController:` (page 1856)
  Removes the given controller of an accessory view.
- `accessoryControllers` (page 1853)
  Returns an array of accessory view controllers belonging to the receiver.

Accessing the NSPrintInfo Object

- `printInfo` (page 1855)
  Returns the `NSPrintInfo` object used when the receiver is run.

Deprecated Methods

- `accessoryView` (page 1853)
  Returns the receiver's accessory view (used to customize the receiver). (Deprecated. Deprecated in Mac OS X v10.5. Use `accessoryControllers` (page 1853) instead.)
- `setAccessoryView:` (page 1857)
  Adds a view object to the receiver. (Deprecated. Deprecated in Mac OS X v10.5. Use `addAccessoryController:` (page 1854) instead.)
- `readPrintInfo` (page 1855) Deprecated in Mac OS X v10.5
  Sets the receiver's values to those stored in the `NSPrintInfo` object used when the receiver is run. (Deprecated. Deprecated in Mac OS X v10.5. This method should not be invoked directly, so there is no replacement.)
- `writePrintInfo` (page 1858) Deprecated in Mac OS X v10.5
  Writes the receiver's values to the `NSPrintInfo` object used when the receiver is run. (Deprecated. Deprecated in Mac OS X v10.5. This method should not be invoked directly, so there is no replacement.)

Class Methods

pageLayout

Returns a newly created `NSPageLayout` object.

```
+ (NSPageLayout *)pageLayout
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
From A View to A Movie
From A View to A Picture
GLSL Showpiece Lite
GLUT
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSPageLayout.h`

Instance Methods

accessoryControllers

Returns an array of accessory view controllers belonging to the receiver.

```
- (NSArray *)accessoryControllers
```

Return Value
The `NSViewController` instances representing the accessory view controllers belonging to the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- addAccessoryController:` (page 1854)
`- removeAccessoryController:` (page 1856)

Declared In
`NSPageLayout.h`

accessoryView

Returns the receiver's accessory view (used to customize the receiver). (Deprecated in Mac OS X v10.5. Deprecated in Mac OS X v10.5. Use `accessoryControllers` (page 1853) instead.)

```
- (NSView *)accessoryView
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

See Also
- `setAccessoryView:` (page 1857)

Declared In
`NSPageLayout.h`

addAccessoryController:

Adds the given controller of an accessory view to be presented in the page setup panel.

```
- (void)addAccessoryController:(NSViewController *)accessoryController
```

Parameters

accessoryController
The controller to add.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `removeAccessoryController:` (page 1856)
- `accessoryControllers` (page 1853)

Declared In
`NSPageLayout.h`

beginSheetWithPrintInfo:modalForWindow:delegate:didEndSelector:contextInfo:

Presents a page setup sheet for the given `NSPrintInfo` object, document-modal relative to the given window.

```
- (void)beginSheetWithPrintInfo:(NSPrintInfo *)printInfo modalForWindow:(NSWindow
    *)docWindow delegate:(id)delegate didEndSelector:(SEL)didEndSelector
   contextInfo:(void *)contextInfo
```

Parameters

printInfo
The `NSPrintInfo` object to use.

docWindow
The window to which the sheet is attached.

delegate
The delegate to which *didEndSelector* is sent. Can be `nil`.

didEndSelector
The selector sent to the delegate. Can be `nil`.

contextInfo
Context information object passed with *didEndSelector*.

Discussion
The *didEndSelector* argument must have the same signature as:

```
- (void)pageLayoutDidEnd:(NSPageLayout *)pageLayout returnCode:(int)returnCode
  contextInfo: (void *)contextInfo;
```

The value passed as *returnCode* is either `NSCancelButton` or `NSOKButton`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
From A View to A Movie
From A View to A Picture
GLSL Showpiece Lite
GLUT
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSPageLayout.h`

printInfo

Returns the `NSPrintInfo` object used when the receiver is run.

```
- (NSPrintInfo *)printInfo
```

Discussion
The NSPrintInfo object is set using the `beginSheetWithPrintInfo:modalForWindow:delegate:didEndSelector:contextInfo:` (page 1854) or `runModalWithPrintInfo:` (page 1857) method. The shared NSPrintInfo object is used if the receiver is run using `runModal` (page 1856).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `readPrintInfo` (page 1855)
- `writePrintInfo` (page 1858)

Declared In
`NSPageLayout.h`

readPrintInfo

Sets the receiver's values to those stored in the `NSPrintInfo` object used when the receiver is run. (Deprecated in Mac OS X v10.5. Deprecated in Mac OS X v10.5. This method should not be invoked directly, so there is no replacement.)

```
- (void)readPrintInfo
```

Discussion
Do not invoke this method directly; it is invoked automatically before the receiver is displayed.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

See Also
- `printInfo` (page 1855)
- `writePrintInfo` (page 1858)
- `runModal` (page 1856)
- `runModalWithPrintInfo:` (page 1857)

Declared In
`NSPageLayout.h`

removeAccessoryController:

Removes the given controller of an accessory view.

```
- (void)removeAccessoryController:(NSViewController *)accessoryController
```

Parameters
accessoryController
The controller to remove.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `addAccessoryController:` (page 1854)
- `accessoryControllers` (page 1853)

Declared In
`NSPageLayout.h`

runModal

Displays the receiver and begins the modal loop using the shared `NSPrintInfo` object.

```
- (NSInteger)runModal
```

Return Value
`NSCancelButton` if the user clicks the Cancel button; otherwise, `NSOKButton`.

Discussion
The receiver's values are recorded in the shared `NSPrintInfo` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `runModalWithPrintInfo:` (page 1857)

Declared In
NSPageLayout.h

runModalWithPrintInfo:

Displays the receiver and begins the modal loop using the given NSPrintInfo object.

```
- (NSInteger)runModalWithPrintInfo:(NSPrintInfo *)printInfo
```

Parameters
printInfo
The NSPrintInfo object to use.

Return Value
NSCancelButton if the user clicks the Cancel button; otherwise, NSOKButton.

Discussion
The receiver's values are recorded in *printInfo*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- runModal (page 1856)

Declared In
NSPageLayout.h

setAccessoryView:

Adds a view object to the receiver. (Deprecated in Mac OS X v10.5. Deprecated in Mac OS X v10.5. Use addAccessoryController: (page 1854) instead.)

```
- (void)setAccessoryView:(NSView *)aView
```

Discussion
Invoke this method to add a custom view containing your controls. *aView* is added to the receiver's Settings popup menu with your application's name as its menu item. The receiver is automatically resized to accommodate *aView*. This method can be invoked repeatedly to change the accessory view depending on the situation. If *aView* is nil, then the receiver's current accessory view, if any, is removed.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

See Also
- accessoryView (page 1853)

Declared In
NSPageLayout.h

writePrintInfo

Writes the receiver's values to the `NSPrintInfo` object used when the receiver is run. (Deprecated in Mac OS X v10.5. Deprecated in Mac OS X v10.5. This method should not be invoked directly, so there is no replacement.)

```
- (void)writePrintInfo
```

Discussion

Do not invoke this method directly; it is invoked automatically when the receiver is dismissed.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

See Also

- `printInfo` (page 1855)
- `readPrintInfo` (page 1855)
- `runModal` (page 1856)
- `runModalWithPrintInfo:` (page 1857)

Declared In

`NSPageLayout.h`

NSPanel Class Reference

Inherits from	NSWindow : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSWindow) NSAnimatablePropertyContainer (NSWindow) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSPanel.h
Companion guide	Window Programming Guide
Related sample code	From A View to A Movie PDF Annotation Editor QTAudioExtractionPanel Quartz Composer WWDC 2005 TextEdit Sketch-112

Overview

The `NSPanel` class implements a special kind of window (known as a **panel**), typically performing an auxiliary function.

For details about how panels work (especially to find out how their behavior differs from window behavior), see How Panels Work.

Tasks

Configuring Panels

`- isFloatingPanel` (page 1860)
Indicates whether the receiver is a floating panel.

`- setFloatingPanel:` (page 1861)
Controls whether the receiver floats above normal windows.

- `becomesKeyOnlyIfNeeded` (page 1860)
  Indicates whether the receiver becomes the key window only when needed.
- `setBecomesKeyOnlyIfNeeded:` (page 1861)
  Specifies whether the receiver becomes the key window only when needed.
- `worksWhenModal` (page 1862)
  Indicates whether the receiver receives keyboard and mouse events even when some other window is being run modally.
- `setWorksWhenModal:` (page 1862)
  Specifies whether the receiver receives keyboard and mouse events even when some other window is being run modally.

Instance Methods

becomesKeyOnlyIfNeeded

Indicates whether the receiver becomes the key window only when needed.

```
- (BOOL)becomesKeyOnlyIfNeeded
```

Return Value
`YES` when the panel becomes the key window only when needed, `NO` otherwise.

Discussion
By default, this attribute is set to `NO`, indicating that the panel becomes key as other windows do.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBecomesKeyOnlyIfNeeded:` (page 1861)
- `needsPanelToBecomeKey` (page 3191) (`NSView`)

Declared In
`NSPanel.h`

isFloatingPanel

Indicates whether the receiver is a floating panel.

```
- (BOOL)isFloatingPanel
```

Return Value
`YES` when the receiver is a floating panel, `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFloatingPanel:` (page 1861)

- level (page 3343) (NSWindow)

Declared In
NSPanel.h

setBecomesKeyOnlyIfNeeded:

Specifies whether the receiver becomes the key window only when needed.

- (void)setBecomesKeyOnlyIfNeeded:(BOOL)*becomesKeyOnlyIfNeeded*

Parameters

becomesKeyOnlyIfNeeded

YES makes the panel become the key window only when keyboard input is required. NO makes the panel become key when it's clicked.

Discussion
This behavior is not set by default. You should consider setting it only if most user interface elements in the panel aren't text fields, and if the choices that can be made by entering text can also be made in another way (such as by clicking an item in a list).

If the receiver is a non-activating panel, then it becomes key only if the hit view returns YES from needsPanelToBecomeKey (page 3191). This way, a non-activating panel can control whether it takes keyboard focus.

Availability
Available in Mac OS X v10.0 and later.

See Also
- becomesKeyOnlyIfNeeded (page 1860)
- needsPanelToBecomeKey (page 3191) (NSView)

Declared In
NSPanel.h

setFloatingPanel:

Controls whether the receiver floats above normal windows.

- (void)setFloatingPanel:(BOOL)*floatingPanel*

Parameters

floatingPanel

YES to make the receiver a floating panel (NSFloatingWindowLevel). NO to make the receiver behave like a normal window (NSNormalWindowLevel).

Discussion
By default, panels do not float above other windows. It's appropriate for an panel to float above other windows only if all of the following conditions are true:

- It's small enough not to obscure whatever is behind it.
- It's oriented more to the mouse than to the keyboard—that is, if it doesn't become the key window or becomes so only when needed.

- It needs to remain visible while the user works in the application's normal windows—for example, if the user must frequently move the cursor back and forth between a normal window and the panel (such as a tool palette), or if the panel gives information relevant to the user's actions in a normal window.
- It hides when the application is deactivated (the default behavior for panels).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isFloatingPanel` (page 1860)
- `setLevel:` (page 3387) (NSWindow)

Related Sample Code
From A View to A Movie
ImageBrowser
ImageKitDemo

Declared In
`NSPanel.h`

setWorksWhenModal:

Specifies whether the receiver receives keyboard and mouse events even when some other window is being run modally.

```
- (void)setWorksWhenModal:(BOOL)worksWhenModal
```

Parameters
worksWhenModal
`YES` to make the panel receive events even during a modal loop or session. `NO` to prevent the panel from receiving events while a modal loop or session is running.

Discussion
See "How Modal Windows Work" for more information on modal windows and panels.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `worksWhenModal` (page 1862)
- `runModalForWindow:` (page 163) (`NSApplication`)
- `runModalSession:` (page 164) (`NSApplication`)

Declared In
`NSPanel.h`

worksWhenModal

Indicates whether the receiver receives keyboard and mouse events even when some other window is being run modally.

```
- (BOOL)worksWhenModal
```

Return Value
`YES` when the receiver receives keyboard and mouse events even when some other window is being run modally, `NO` otherwise.

Discussion
By default, this attribute is set to `NO`, indicating a panel's ineligibility for events during a modal loop or session. See "How Modal Windows Work" for more information on modal windows and panels.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setWorksWhenModal:` (page 1862)
- `runModalForWindow:` (page 163) (`NSApplication`)
- `runModalSession:` (page 164) (`NSApplication`)

Declared In
`NSPanel.h`

Constants

Alert Panel Return Values

These constants define values returned by the `NSRunAlertPanel` (page 3998) function and by the `NSApplication` method `runModalSession:` (page 164) when the modal session is run with an `NSPanel` provided by the `NSGetAlertPanel` (page 3983) function.

```
enum {
   NSAlertDefaultReturn = 1,
   NSAlertAlternateReturn = 0,
   NSAlertOtherReturn = -1,
   NSAlertErrorReturn = -2
};
```

Constants

`NSAlertDefaultReturn`
The user pressed the default button.
Available in Mac OS X v10.0 and later.
Declared in `NSPanel.h`.

`NSAlertAlternateReturn`
The user pressed the alternate button.
Available in Mac OS X v10.0 and later.
Declared in `NSPanel.h`.

`NSAlertOtherReturn`
The user pressed a second alternate button.
Available in Mac OS X v10.0 and later.
Declared in `NSPanel.h`.

`NSAlertErrorReturn`
: The alert cannot identify the reason it was closed; it may have been closed by an external source or by a button other than those listed above.

  Available in Mac OS X v10.0 and later.

  Declared in `NSPanel.h`.

Declared In
`NSPanel.h`

Modal Panel Return Values

These constants define the possible return values for such methods as the `runModal...` methods of the `NSOpenPanel` class, which tell which button (OK or Cancel) the user has clicked on an open panel.

```
enum {
   NSOKButton = 1,
   NSCancelButton = 0
};
```

Constants
`NSCancelButton`
: The Cancel button

  Available in Mac OS X v10.0 and later.

  Declared in `NSPanel.h`.

`NSOKButton`
: The OK button

  Available in Mac OS X v10.0 and later.

  Declared in `NSPanel.h`.

Declared In
`NSPanel.h`

Style Masks

The `NSPanel` class defines the following constants for panel styles:

```
enum {
   NSUtilityWindowMask = 1 << 4,
   NSDocModalWindowMask = 1 << 6,
   NSNonactivatingPanelMask = 1 << 7
   NSHUDWindowMask = 1 << 13
};
```

Constants
`NSDocModalWindowMask`
: The panel is created as a modal sheet.

  Available in Mac OS X v10.0 and later.

  Declared in `NSPanel.h`.

`NSUtilityWindowMask`
The panel is created as a floating window.

Available in Mac OS X v10.0 and later.

Declared in `NSPanel.h`.

`NSNonactivatingPanelMask`
The panel can receive keyboard input without activating the owning application.

Valid only for an instance of `NSPanel` or its subclasses; not valid for a window.

Available in Mac OS X v10.2 and later.

Declared in `NSPanel.h`.

`NSHUDWindowMask`
The panel is created as a transparent panel (sometimes called a "heads-up display").

Valid only for an instance of `NSPanel` or its subclasses; not valid for a window.

Using the C bitwise OR operator, `NSHUDWindowMask` can be combined with other style masks (some of which are documented in `Window_Style_Masks` (page 3411)) with the following results:

`NSBorderlessWindowMask`
Borderless window with transparent panel transparency and window level.

`NSTitledWindowMask | NSUtilityWindowMask`
Titled window with transparent panel transparency and window level. This combination can be additionally combined with any of the following:

`NSClosableWindowMask`
Titled window with transparent panel close box, transparency, and window level.

`NSResizableWindowMask`
Titled window with transparent panel resize corner, transparency, and window level.

`NSNonactivatingPanelMask`
No effect on appearance, but owning application is not necessarily active when this window is the key window.

The following constants cannot be combined with `NSHUDWindowMask`: `NSMiniaturizableWindowMask`, `NSTexturedBackgroundWindowMask`, `NSDocModalWindowMask`, and `NSUnifiedTitleAndToolbarWindowMask`.

Available in Mac OS X v10.5 and later.

Declared in `NSPanel.h`.

Declared In
`NSPanel.h`

NSParagraphStyle Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSMutableCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSParagraphStyle.h
Companion guide	Rulers and Paragraph Styles
Related sample code	Cocoa Tips and Tricks FilterDemo iSpend Quartz Composer WWDC 2005 TextEdit TipWrapper

Overview

`NSParagraphStyle` and its subclass `NSMutableParagraphStyle` encapsulate the paragraph or ruler attributes used by the `NSAttributedString` classes. Instances of these classes are often referred to as paragraph style objects or, when no confusion will result, paragraph styles.

The mutable subclass of `NSParagraphStyle` is NSMutableParagraphStyle.

Adopted Protocols

NSCoding

- `- encodeWithCoder:`
- `- initWithCoder:`

NSCopying

- `- copyWithZone:`

NSMutableCopying

```
- mutableCopyWithZone:
```

Tasks

Creating an NSParagraphStyle

`+ defaultParagraphStyle` (page 1869)
Returns the default paragraph style.

Accessing Style Information

`- alignment` (page 1870)
Returns the text alignment of the receiver.

`- firstLineHeadIndent` (page 1872)
Returns the indentation of the first line of the receiver.

`- headIndent` (page 1872)
Returns the indentation of the receiver's lines other than the first.

`- tailIndent` (page 1877)
Returns the trailing indentation of the receiver.

`- tabStops` (page 1876)
Returns the receiver's tab stops.

`- defaultTabInterval` (page 1871)
Returns the document-wide default tab interval.

`- lineHeightMultiple` (page 1873)
Returns the line height multiple.

`- maximumLineHeight` (page 1874)
Returns the receiver's maximum line height.

`- minimumLineHeight` (page 1875)
Returns the receiver's minimum height.

`- lineSpacing` (page 1874)
Returns the space between lines in the receiver (commonly known as leading).

`- paragraphSpacing` (page 1875)
Returns the space after the end of the paragraph.

`- paragraphSpacingBefore` (page 1876)
Returns the distance between the paragraph's top and the beginning of its text content.

Getting Text Block and List Information

`- textBlocks` (page 1877)
Returns an array specifying the text blocks containing the paragraph.

- `textLists` (page 1877)
  Returns an array specifying the text lists containing the paragraph.

Getting Line Breaking Information

- `lineBreakMode` (page 1873)
  Returns the mode that should be used to break lines in the receiver.
- `hyphenationFactor` (page 1873)
  Returns the paragraph's threshold for hyphenation.
- `tighteningFactorForTruncation` (page 1878)
  Returns the threshold for using tightening as an alternative to truncation.

Getting HTML Header Level

- `headerLevel` (page 1872)
  Specifies whether the paragraph is to be treated as a header for purposes of HTML generation.

Writing Direction

+ `defaultWritingDirectionForLanguage:` (page 1870)
  Returns the default writing direction for the specified language.
- `baseWritingDirection` (page 1871)
  Returns the base writing direction for the receiver.

Class Methods

defaultParagraphStyle

Returns the default paragraph style.

```
+ (NSParagraphStyle *)defaultParagraphStyle
```

Discussion
The default paragraph style has the following default values:

Subattribute	Default Value
Alignment	`NSNaturalTextAlignment`
Tab stops	12 left-aligned tabs, spaced by 28.0 points
Line break mode	`NSLineBreakByWordWrapping`
All others	0.0

See individual method descriptions for explanations of each subattribute.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Cocoa Tips and Tricks
FilterDemo
iSpend
PDFKitLinker2
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSParagraphStyle.h`

defaultWritingDirectionForLanguage:

Returns the default writing direction for the specified language.

```
+ (NSWritingDirection)defaultWritingDirectionForLanguage:(NSString *)languageName
```

Parameters

languageName

The language specified in ISO language region format. Can be `nil` to return a default writing direction derived from the user's defaults database.

Return Value
The default writing direction.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- baseWritingDirection` (page 1871)
`- setBaseWritingDirection:` (page 1718) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

Instance Methods

alignment

Returns the text alignment of the receiver.

```
- (NSTextAlignment)alignment
```

Return Value
The text alignment.

Discussion
Natural text alignment is realized as left or right alignment depending on the line sweep direction of the first script contained in the paragraph.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlignment:` (page 1718) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

baseWritingDirection

Returns the base writing direction for the receiver.

```
- (NSWritingDirection)baseWritingDirection
```

Return Value
The base writing direction for the receiver.

Availability
Available in Mac OS X v10.2 and later.

See Also
+ `defaultWritingDirectionForLanguage:` (page 1870)
- `setBaseWritingDirection:` (page 1718) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

defaultTabInterval

Returns the document-wide default tab interval.

```
- (CGFloat)defaultTabInterval
```

Return Value
The default tab interval in points. Tabs after the last specified in `tabStops` (page 1876) are placed at integer multiples of this distance (if positive). Default return value is 0.0.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setDefaultTabInterval:` (page 1719) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

firstLineHeadIndent

Returns the indentation of the first line of the receiver.

```
- (CGFloat)firstLineHeadIndent
```

Return Value
The distance in points from the leading margin of a text container to the beginning of the paragraph's first line. This value is always nonnegative.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `headIndent` (page 1872)
- `tailIndent` (page 1877)
- `setFirstLineHeadIndent:` (page 1719) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

headerLevel

Specifies whether the paragraph is to be treated as a header for purposes of HTML generation.

```
- (NSInteger)headerLevel
```

Return Value
Returns 0 (the default value), if the paragraph is not a header, or from 1 through 6 if the paragraph is to be treated as a header.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setHeaderLevel:` (page 1720) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

headIndent

Returns the indentation of the receiver's lines other than the first.

```
- (CGFloat)headIndent
```

Return Value
The distance in points from the leading margin of a text container to the beginning of lines other than the first. This value is always nonnegative.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `firstLineHeadIndent` (page 1872)
- `tailIndent` (page 1877)
- `setHeadIndent:` (page 1720) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

hyphenationFactor

Returns the paragraph's threshold for hyphenation.

```
- (float)hyphenationFactor
```

Return Value
A value between 0.0 and 1.0 inclusive. The default value is 0.0.

Discussion
Hyphenation is attempted when the ratio of the text width (as broken without hyphenation) to the width of the line fragment is less than the hyphenation factor. When the paragraph's hyphenation factor is 0.0, the layout manager's hyphenation factor is used instead. When both are 0.0, hyphenation is disabled.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setHyphenationFactor:` (page 1720) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

lineBreakMode

Returns the mode that should be used to break lines in the receiver.

```
- (NSLineBreakMode)lineBreakMode
```

Return Value
The line break mode to be used laying out the paragraph's text.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setLineBreakMode:` (page 1721) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

lineHeightMultiple

Returns the line height multiple.

```
- (CGFloat)lineHeightMultiple
```

Return Value
The line height multiple. The natural line height of the receiver is multiplied by this factor (if positive) before being constrained by minimum and maximum line height. Default return value is 0.0.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `maximumLineHeight` (page 1874)
- `minimumLineHeight` (page 1875)
- `setLineHeightMultiple:` (page 1722) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

lineSpacing

Returns the space between lines in the receiver (commonly known as leading).

```
- (CGFloat)lineSpacing
```

Return Value
The space in points added between lines within the paragraph. This value is always nonnegative.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `maximumLineHeight` (page 1874)
- `minimumLineHeight` (page 1875)
- `paragraphSpacing` (page 1875)
- `setLineSpacing:` (page 1722) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

maximumLineHeight

Returns the receiver's maximum line height.

```
- (CGFloat)maximumLineHeight
```

Return Value
The maximum height in points that any line in the receiver will occupy, regardless of the font size or size of any attached graphic. This value is always nonnegative. The default value is 0.

Discussion
Glyphs and graphics exceeding this height will overlap neighboring lines; however, a maximum height of 0 implies no line height limit. Although this limit applies to the line itself, line spacing adds extra space between adjacent lines.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `minimumLineHeight` (page 1875)
- `lineSpacing` (page 1874)
- `lineHeightMultiple` (page 1873)
- `setMaximumLineHeight:` (page 1722) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

minimumLineHeight

Returns the receiver's minimum height.

```
- (CGFloat)minimumLineHeight
```

Return Value
The minimum height in points that any line in the receiver will occupy, regardless of the font size or size of any attached graphic. This value is always nonnegative.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `maximumLineHeight` (page 1874)
- `lineSpacing` (page 1874)
- `lineHeightMultiple` (page 1873)
- `setMinimumLineHeight:` (page 1723) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

paragraphSpacing

Returns the space after the end of the paragraph.

```
- (CGFloat)paragraphSpacing
```

Return Value
The space in points added at the end of the paragraph to separate it from the following paragraph. This value is always nonnegative.

Discussion
This value is determined by adding the previous paragraph's `paragraphSpacing` and the current paragraph's `paragraphSpacingBefore` (page 1876).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lineSpacing` (page 1874)
- `paragraphSpacingBefore` (page 1876)
- `setParagraphSpacing:` (page 1723) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

paragraphSpacingBefore

Returns the distance between the paragraph's top and the beginning of its text content.

```
- (CGFloat)paragraphSpacingBefore
```

Return Value
The distance in points between the paragraph's top and the beginning of its text content. Default return value is 0.0.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `paragraphSpacing` (page 1875)
- `setParagraphSpacingBefore:` (page 1724) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

tabStops

Returns the receiver's tab stops.

```
- (NSArray *)tabStops
```

Return Value
The `NSTextTab` objects, sorted by location, that define the tab stops for the paragraph style.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `location` (page 2851) (`NSTextTab`)
- `setTabStops:` (page 1724) (`NSMutableParagraphStyle`)
- `addTabStop:` (page 1717) (`NSMutableParagraphStyle`)
- `removeTabStop:` (page 1717) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

tailIndent

Returns the trailing indentation of the receiver.

```
- (CGFloat)tailIndent
```

Return Value
The distance in points from the margin of a text container to the end of lines.

Discussion
If positive, this value is the distance from the leading margin (for example, the left margin in left-to-right text). If 0 or negative, it's the distance from the trailing margin.

For example, a paragraph style designed to fit exactly in a 2-inch wide container has a head indent of 0.0 and a tail indent of 0.0. One designed to fit with a quarter-inch margin has a head indent of 0.25 and a tail indent of –0.25.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `headIndent` (page 1872)
- `firstLineHeadIndent` (page 1872)
- `setTailIndent:` (page 1725) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

textBlocks

Returns an array specifying the text blocks containing the paragraph.

```
- (NSArray *)textBlocks
```

Return Value
An array of the `NSTextTableBlock` objects containing the paragraph, nested from outermost to innermost.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTextBlocks:` (page 1725) (`NSMutableParagraphStyle`)

Related Sample Code
iSpend

Declared In
`NSParagraphStyle.h`

textLists

Returns an array specifying the text lists containing the paragraph.

```
- (NSArray *)textLists
```

Return Value
An array of the `NSTextList` objects containing the paragraph, nested from outermost to innermost.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setTextLists:` (page 1726) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

tighteningFactorForTruncation

Returns the threshold for using tightening as an alternative to truncation.

```
- (float)tighteningFactorForTruncation
```

Return Value
The tightening threshold value. The default value is 0.05.

Discussion
When the line break mode specifies truncation, the text system attempts to tighten intercharacter spacing as an alternative to truncation, provided that the ratio of the text width to the line fragment width does not exceed 1.0 + the tightening factor returned by this method. Otherwise the text is truncated at a location determined by the line break mode.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setTighteningFactorForTruncation:` (page 1726) (`NSMutableParagraphStyle`)

Declared In
`NSParagraphStyle.h`

Constants

NSLineBreakMode

These constants specify what happens when a line is too long for its container.

```
enum {
    NSLineBreakByWordWrapping = 0,
    NSLineBreakByCharWrapping,
    NSLineBreakByClipping,
    NSLineBreakByTruncatingHead,
    NSLineBreakByTruncatingTail,
    NSLineBreakByTruncatingMiddle
};
typedef NSUInteger NSLineBreakMode
```

Constants

`NSLineBreakByWordWrapping`
Wrapping occurs at word boundaries, unless the word itself doesn't fit on a single line.

Available in Mac OS X v10.0 and later.

Declared in `NSParagraphStyle.h`.

`NSLineBreakByCharWrapping`
Wrapping occurs before the first character that doesn't fit.

Available in Mac OS X v10.0 and later.

Declared in `NSParagraphStyle.h`.

`NSLineBreakByClipping`
Lines are simply not drawn past the edge of the text container.

Available in Mac OS X v10.0 and later.

Declared in `NSParagraphStyle.h`.

`NSLineBreakByTruncatingHead`
Each line is displayed so that the end fits in the container and the missing text is indicated by some kind of ellipsis glyph.

Available in Mac OS X v10.0 and later.

Declared in `NSParagraphStyle.h`.

`NSLineBreakByTruncatingTail`
Each line is displayed so that the beginning fits in the container and the missing text is indicated by some kind of ellipsis glyph.

Available in Mac OS X v10.0 and later.

Declared in `NSParagraphStyle.h`.

`NSLineBreakByTruncatingMiddle`
Each line is displayed so that the beginning and end fit in the container and the missing text is indicated by some kind of ellipsis glyph in the middle.

Available in Mac OS X v10.0 and later.

Declared in `NSParagraphStyle.h`.

NSPasteboard Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSPasteboard.h
Companion guides	Pasteboard Programming Guide Drag and Drop Programming Topics for Cocoa Services Implementation Guide
Related sample code	DemoAssistant DemoMonkey iSpend Quartz Composer WWDC 2005 TextEdit Sketch-112

Overview

`NSPasteboard` objects transfer data to and from the pasteboard server. The server is shared by all running applications. It contains data that the user has cut or copied, as well as other data that one application wants to transfer to another. `NSPasteboard` objects are an application's sole interface to the server and to all pasteboard operations.

An `NSPasteboard` object is also used to transfer data between applications and service providers listed in each application's Services menu. The drag pasteboard (`NSDragPboard`) is used to transfer data that is being dragged by the user.

Important: This reference and the *Pasteboard Programming Guide* describe the API and behavior for Mac OS X v10.6 and later. To learn about the behavior on Mac OS X v10.5 and earlier, see "Pasteboard Programming Topics for Cocoa" in the Legacy Reference Library.

On Mac OS X v10.6 and later, a pasteboard can contain multiple items. You can directly write or read any object that implements the `NSPasteboardWriting Protocol Reference` or `NSPasteboardReading Protocol Reference` protocol respectively. This allows you to write and read common items such as URLs, colors, images, strings, attributed strings, and sounds without an intermediary object. Your custom classes can also implement these protocols for use with the pasteboard.

The writing methods available on Mac OS X v10.5 and earlier all operate on what is conceptually the first item on the pasteboard. They accept UTIs and pboard type strings. In a future release they may take only UTIs. On Mac OS X v10.6 and later, the writing methods such as `setData:forType:` (page 1898) still provide a convenient means of writing to the first pasteboard item, without having to create the first pasteboard item. For example, instead of

```
[pboard clearContents];
NSPasteboardItem *item = [[[NSPasteboardItem alloc] init] autorelease];
[item setData:data forType:type];
[pboard writeObjects:[NSArray arrayWithObject:item]];
```

you can write:

```
[pboard clearContents];
[pboard setData:data forType:type];
```

In addition to being more compact, this is also a little more efficient—it avoids the need to create a separate pasteboard item and array.

Tasks

Creating and Releasing a Pasteboard

`+ generalPasteboard` (page 1884)
Returns the general `NSPasteboard` object.

`+ pasteboardByFilteringData:ofType:` (page 1885)
Creates and returns a new pasteboard with a unique name that supplies the specified data in as many types as possible given the available filter services.

`+ pasteboardByFilteringFile:` (page 1885)
Creates and returns a new pasteboard with a unique name that supplies the specified file data in as many types as possible given the available filter services.

`+ pasteboardByFilteringTypesInPasteboard:` (page 1886)
Creates and returns a new pasteboard with a unique name that supplies the specified pasteboard data in as many types as possible given the available filter services.

`+ pasteboardWithName:` (page 1886)
Returns the pasteboard with the specified name.

`+ pasteboardWithUniqueName` (page 1887)
Creates and returns a new pasteboard with a name that is guaranteed to be unique with respect to other pasteboards on the computer.

`- releaseGlobally` (page 1898)
Releases the receiver's resources in the pasteboard server.

Writing Data

`- clearContents` (page 1891)
Clears the existing contents of the pasteboard

- `writeObjects:` (page 1903)
  Writes an array of objects to the receiver.
- `setData:forType:` (page 1898)
  Sets the given data as the representation for the specified type for the first item on the receiver.
- `setPropertyList:forType:` (page 1899)
  Sets the given property list as the representation for the specified type for the first item on the receiver.
- `setString:forType:` (page 1900)
  Sets the given string as the representation for the specified type for the first item on the receiver.

Reading Data

- `readObjectsForClasses:options:` (page 1897)
  Reads from the receiver objects that best match the specified array of classes.
- `pasteboardItems` (page 1895)
  Returns all the items held by the receiver.
- `indexOfPasteboardItem:` (page 1894)
  Returns the index of the specified pasteboard item.
- `dataForType:` (page 1892)
  Returns the data for the specified type from the first item in the receiver that contains the type.
- `propertyListForType:` (page 1895)
  Returns the property list for the specified type from the first item in the receiver that contains the type.
- `stringForType:` (page 1901)
  Returns a concatenation of the strings for the specified type from all the items in the receiver that contain the type.

Validating Contents

- `availableTypeFromArray:` (page 1889)
  Scans the specified types for a type that the receiver supports.
- `canReadItemWithDataConformingToTypes:` (page 1890)
  Returns a Boolean value that indicates whether the receiver contains any items that conform to the specified UTIs.
- `canReadObjectForClasses:options:` (page 1890)
  Returns a Boolean value that indicates whether the receiver contains any items that can be represented as an instance of any class in a given array.
- `types` (page 1901)
  Returns an array of the receiver's supported data types.
+ `typesFilterableTo:` (page 1887)
  Returns the data types that can be converted to the specified type using the available filter services.

Getting Information About a Pasteboard

- `name` (page 1894)
  Returns the receiver's name.
- `changeCount` (page 1891)
  Returns the receiver's change count.

Writing Data (Mac OS X v10.5 and Earlier)

These methods all operate on what is conceptually the first item on the pasteboard. They accept UTIs and pboard type strings. In a future release they may take only UTIs.

- `declareTypes:owner:` (page 1893)
  Prepares the receiver for a change in its contents by declaring the new types of data it will contain and a new owner.
- `addTypes:owner:` (page 1888)
  Adds promises for the specified types to the first pasteboard item.
- `writeFileContents:` (page 1902)
  Writes the contents of the specified file to the pasteboard.
- `writeFileWrapper:` (page 1902)
  Writes the serialized contents of the specified file wrapper to the pasteboard.

Reading Data (Mac OS X v10.5 and Earlier)

These methods all operate on what is conceptually the first item on the pasteboard. They accept UTIs and pboard type strings. In a future release they may take only UTIs.

- `readFileContentsType:toFile:` (page 1896)
  Reads data representing a file's contents from the receiver and writes it to the specified file.
- `readFileWrapper` (page 1896)
  Reads data representing a file's contents from the receiver and returns it as a file wrapper.

Class Methods

generalPasteboard

Returns the general `NSPasteboard` object.

```
+ (NSPasteboard *)generalPasteboard
```

Return Value
The general pasteboard.

Discussion
Invokes `pasteboardWithName:` (page 1886) to obtain the pasteboard.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DemoMonkey
PDFKitLinker2
RGB Image
Sketch+Accessibility
Sketch-112

Declared In
NSPasteboard.h

pasteboardByFilteringData:ofType:

Creates and returns a new pasteboard with a unique name that supplies the specified data in as many types as possible given the available filter services.

```
+ (NSPasteboard *)pasteboardByFilteringData:(NSData *)data ofType:(NSString *)type
```

Parameters

data
The data to be placed on the pasteboard.

type
The type of data in the *data* parameter.

Return Value
The new pasteboard object.

Discussion
The returned pasteboard also declares data of the supplied *type*.

No filter service is invoked until the data is actually requested, so invoking this method is reasonably inexpensive.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPasteboard.h

pasteboardByFilteringFile:

Creates and returns a new pasteboard with a unique name that supplies the specified file data in as many types as possible given the available filter services.

```
+ (NSPasteboard *)pasteboardByFilteringFile:(NSString *)filename
```

Parameters

filename
The filename to put on the pasteboard.

Return Value
The new pasteboard object.

Discussion
No filter service is invoked until the data is actually requested, so invoking this method is reasonably inexpensive.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPasteboard.h

pasteboardByFilteringTypesInPasteboard:

Creates and returns a new pasteboard with a unique name that supplies the specified pasteboard data in as many types as possible given the available filter services.

```
+ (NSPasteboard *)pasteboardByFilteringTypesInPasteboard:(NSPasteboard *)pasteboard
```

Parameters

pasteboard
The original pasteboard object.

Return Value
The new pasteboard object. This method returns the object in the *pasteboard* parameter if the pasteboard was returned by one of the `pasteboardByFiltering...` methods. This prevents a pasteboard from being expanded multiple times.

Discussion
This process can be thought of as expanding the pasteboard, because the new pasteboard generally contains more representations of the data than *pasteboard*.

This method only returns the original types and the types that can be created as a result of a single filter; the pasteboard does not have defined types that are the result of translation by multiple filters.

No filter service is invoked until the data is actually requested, so invoking this method is reasonably inexpensive.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPasteboard.h

pasteboardWithName:

Returns the pasteboard with the specified name.

```
+ (NSPasteboard *)pasteboardWithName:(NSString *)name
```

Parameters

name

The name of the pasteboard. The names of standard pasteboards are given in "Pasteboard Names" (page 1903).

Return Value
The pasteboard associated with the given name, or a new `NSPasteboard` object if the application does not yet have a pasteboard object for the specified name.

Discussion
Other names can be assigned to create private pasteboards for other purposes.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ClipboardViewer
CocoaDragAndDrop
FunHouse
PDFKitLinker2
ZipBrowser

Declared In
`NSPasteboard.h`

pasteboardWithUniqueName

Creates and returns a new pasteboard with a name that is guaranteed to be unique with respect to other pasteboards on the computer.

```
+ (NSPasteboard *)pasteboardWithUniqueName
```

Return Value
The new pasteboard object.

Discussion
This method is useful for applications that implement their own interprocess communication using pasteboards.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
PhotoSearch

Declared In
`NSPasteboard.h`

typesFilterableTo:

Returns the data types that can be converted to the specified type using the available filter services.

```
+ (NSArray *)typesFilterableTo:(NSString *)type
```

Parameters

type

The target data type.

Return Value

An array of `NSString` objects containing the types that can be converted to the target data type.

Discussion

The array also contains the original type.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSPasteboard.h`

Instance Methods

addTypes:owner:

Adds promises for the specified types to the first pasteboard item.

```
- (NSInteger)addTypes:(NSArray *)newTypes owner:(id)newOwner
```

Parameters

newTypes

An array of `NSString` objects, each of which specifies a type of data that can be provided to the pasteboard.

newOwner

The object that provides the data for the specified types.

If the data for those types is provided immediately, the owner can be `nil`. If the data for the added types will be provided lazily when requested from the pasteboard, an owner object must be provided that implements the `-pasteboard:provideDataForType:` method of the `NSPasteboardOwner` informal protocol.

Return Value

The new change count, or `0` if there was an error adding the types.

Discussion

This method adds promises for the specified types to the first pasteboard item.

You use this methods to declare additional types of data for the first pasteboard item in the receiver. You can also use it to replace existing types added by a previous `declareTypes:owner:` (page 1893) or `addTypes:owner:` message.

The *newTypes* parameter specifies the types of data you are promising to the pasteboard. The types should be ordered according to the preference of the source application, with the most preferred type coming first (typically, the richest representation). New types are added to the end of the list containing any existing types, if any.

If you specify a type that has already been declared, this method replaces the owner of that type with the value in `newOwner`. In addition, any data already already written to the pasteboard for that type is removed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- changeCount (page 1891)

Related Sample Code
CocoaDragAndDrop
DemoMonkey
RGB Image
RGB ValueTransformers

Declared In
NSPasteboard.h

availableTypeFromArray:

Scans the specified types for a type that the receiver supports.

```
- (NSString *)availableTypeFromArray:(NSArray *)types
```

Parameters

types

An array of NSString objects specifying the pasteboard types your application supports, in preferred order.

Return Value
The first pasteboard type in *types* that is available on the pasteboard, or nil if the receiver does not contain any of the types in *types*.

Discussion
You use this method to determine the best representation available on the pasteboard. For example, if your application supports RTFD, RTF, and string data, then you might invoke the method as follows:

```
NSArray *supportedTypes =
    [NSArray arrayWithObjects: NSRTFDPboardType, NSRTFPboardType,
NSStringPboardType, nil];
NSString *bestType = [[NSPasteboard generalPasteboard]
    availableTypeFromArray:supportedTypes];
```

If the pasteboard contains RTF and string data, then bestType would contain NSRTFPboardType. If the pasteboard contains none of the types in supportedTypes, then bestType would be nil.

You must send a types (page 1901) or availableTypeFromArray: message before reading any data from an NSPasteboard object. If you need to see if a type in the returned array matches a type string you have stored locally, use the isEqualToString: method to perform the comparison.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
GeekGameBoard
GLUT
MP3 Player

Sketch+Accessibility
Sketch-112

Declared In
NSPasteboard.h

canReadItemWithDataConformingToTypes:

Returns a Boolean value that indicates whether the receiver contains any items that conform to the specified UTIs.

```
- (BOOL)canReadItemWithDataConformingToTypes:(NSArray *)types
```

Parameters

types

An array of NSString objects containing UTIs.

Return Value
YES if the receiver contains any items that conform to the UTIs specified in *types*, otherwise NO.

Availability
Available in Mac OS X v10.6 and later.

See Also
- canReadObjectForClasses:options: (page 1890)
- readObjectsForClasses:options: (page 1897)

Declared In
NSPasteboard.h

canReadObjectForClasses:options:

Returns a Boolean value that indicates whether the receiver contains any items that can be represented as an instance of any class in a given array.

```
- (BOOL)canReadObjectForClasses:(NSArray *)classArray
    options:(NSDictionary *)options
```

Parameters

classArray

An array of class objects.

Classes in the array must conform to the NSPasteboardReading Protocol Reference protocol.

options

A dictionary that specifies options to refine the search for pasteboard items, for example to restrict the search to file URLs with particular content types. For valid dictionary keys, see "Pasteboard Reading Options" (page 1910).

Return Value
YES if the receiver contains any items that can be represented as an instance of a class specified in *classArray*, otherwise NO.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `canReadItemWithDataConformingToTypes:` (page 1890)
- `readObjectsForClasses:options:` (page 1897)

Related Sample Code
DemoMonkey

Declared In
`NSPasteboard.h`

changeCount

Returns the receiver's change count.

```
- (NSInteger)changeCount
```

Return Value
The change count of the receiver.

Discussion
The change count starts at zero when a client creates the receiver and becomes the first owner. The change count subsequently increments each time the pasteboard ownership changes.

The change count is also returned from `clearContents` (page 1891) and `declareTypes:owner:` (page 1893). You can therefore record the change count at the time that you take ownership of the pasteboard and later compare it with the value returned from `changeCount` to determine whether you still have ownership.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `clearContents` (page 1891)
- `declareTypes:owner:` (page 1893)

Related Sample Code
GLUT
Sketch+Accessibility
Sketch-112

Declared In
`NSPasteboard.h`

clearContents

Clears the existing contents of the pasteboard

```
- (NSInteger)clearContents
```

Return Value
The change count of the receiver.

Discussion
Clears the existing contents of the pasteboard, preparing it for new contents. This is the first step in providing data on the pasteboard.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
DemoMonkey
IconCollection

Declared In
`NSPasteboard.h`

dataForType:

Returns the data for the specified type from the first item in the receiver that contains the type.

```
- (NSData *)dataForType:(NSString *)dataType
```

Parameters

dataType

The type of data you want to read from the pasteboard. This value should be one of the types returned by `types` (page 1901) or `availableTypeFromArray:` (page 1889).

Return Value
A data object containing the data for the specified type from the first item in the receiver that contains the type, or `nil` if the contents of the pasteboard changed since they were last checked.

This method may also return `nil` if the pasteboard server cannot supply the data in time—for example, if the pasteboard's owner is slow in responding to a `pasteboard:provideDataForType:` message and the interprocess communication times out.

Discussion
Errors other than a timeout raise a `NSPasteboardCommunicationException`.

If `nil` is returned, the application should put up a panel informing the user that it was unable to carry out the paste operation.

Special Considerations

For standard text data types such as string, RTF, and RTFD, the text data from each item is returned as one combined result separated by newlines.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setData:forType:` (page 1898)

Related Sample Code
iSpend

QTMetadataEditor
RGB Image
Sketch+Accessibility
Sketch-112

Declared In
NSPasteboard.h

declareTypes:owner:

Prepares the receiver for a change in its contents by declaring the new types of data it will contain and a new owner.

```
- (NSInteger)declareTypes:(NSArray *)newTypes owner:(id)newOwner
```

Parameters

newTypes

An array of NSString objects that specify the types of data that may be added to the new pasteboard. The types should be ordered according to the preference of the source application, with the most preferred type coming first (typically, the richest representation).

newOwner

The object responsible for writing data to the pasteboard, or nil if you provide data for all types immediately. If you specify a *newOwner* object, it must support all of the types declared in the *newTypes* parameter and must remain valid for as long as the data is *promised* on the pasteboard.

Return Value
The receiver's new change count.

Discussion
This method is the equivalent of invoking clearContents (page 1891), implicitly writing the first pasteboard item, and then callingaddTypes:owner: (page 1888) to promise types for the first pasteboard item.

Mac OS X v10.5 and earlier: In Mac OS X v10.5 and earlier, this method is the first step in writing data to the pasteboard and must precede the messages that actually write the data. A declareTypes:owner: message essentially changes the contents of the receiver: It invalidates the current contents of the receiver and increments its change count.

Special Considerations

In general, you should not use this method with writeObjects: (page 1903), since writeObjects: will always write additional items to the pasteboard, and will not affect items already on the pasteboard, including the item implicitly created by this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- clearContents (page 1891)
- addTypes:owner: (page 1888)
- changeCount (page 1891)
- pasteboard:provideDataForType: (NSPasteboardOwner Informal Protocol)

Related Sample Code
GLUT
PDFKitLinker2
QTMetadataEditor
Sketch-112
With and Without Bindings

Declared In
`NSPasteboard.h`

indexOfPasteboardItem:

Returns the index of the specified pasteboard item.

```
- (NSUInteger)indexOfPasteboardItem:(NSPasteboardItem *)pasteboardItem
```

Parameters
pasteboardItem
A pasteboard item.

Return Value
The index of the specified pasteboard item. If *pasteboardItem* has not been added to any pasteboard, or is owned by another pasteboard, returns `NSNotFound`.

Discussion
An item's index in the pasteboard is useful for a pasteboard item data provider that has promised data for multiple items, to be able to easily match the pasteboard item to an array of source data from which to derive the promised data.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSPasteboard.h`

name

Returns the receiver's name.

```
- (NSString *)name
```

Return Value
The name of the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ pasteboardWithName:` (page 1886)

Declared In
`NSPasteboard.h`

pasteboardItems

Returns all the items held by the receiver.

```
- (NSArray *)pasteboardItems
```

Return Value
Returns all the items held by the receiver, or `nil` if there is an error retrieving pasteboard items.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- readObjectsForClasses:options:` (page 1897)

Declared In
`NSPasteboard.h`

propertyListForType:

Returns the property list for the specified type from the first item in the receiver that contains the type.

```
- (id)propertyListForType:(NSString *)dataType
```

Parameters
dataType
The pasteboard data type containing the property-list data.

Return Value
The property list for the specified type from the first item in the receiver that contains the type. This object consists of `NSArray`, `NSData`, `NSDictionary`, or `NSString` objects—or any combination thereof.

Discussion
This method invokes the `dataForType:` (page 1892) method.

Special Considerations

You must send `types` (page 1901) or `availableTypeFromArray:` (page 1889) before invoking `propertyListForType:`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setPropertyList:forType:` (page 1899)

Related Sample Code
DictionaryController
QTMetadataEditor
Sketch+Accessibility
Sketch-112
SourceView

Declared In
`NSPasteboard.h`

readFileContentsType:toFile:

Reads data representing a file's contents from the receiver and writes it to the specified file.

```
- (NSString *)readFileContentsType:(NSString *)type toFile:(NSString *)filename
```

Parameters

type

The pasteboard data type to read. You should generally specify a value for this parameter. If you specify `nil`, the filename extension (in combination with the `NSCreateFileContentsPboardType` function) is used to determine the type.

filename

The file to receive the pasteboard data.

Return Value

The name of the file into which the data was actually written.

Discussion

Data of any file contents type should only be read using this method. If data matching the specified type is not found on the pasteboard, data of type `NSFileContentsPboardType` is requested.

Mac OS X v10.6 and later: In Mac OS X v10.5 and earlier, the file contents pboard type allowed you to synthesize a pboard type for a file's contents based on the file's extension. In Mac OS X v10.5 and later, using the UTI of a file to represent its contents now replaces this functionality.

Special Considerations

You must send an `availableTypeFromArray:` (page 1889) or `types` (page 1901) message before invoking `readFileContentsType:toFile:`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- writeFileContents:` (page 1902)

Declared In

`NSPasteboard.h`

readFileWrapper

Reads data representing a file's contents from the receiver and returns it as a file wrapper.

```
- (NSFileWrapper *)readFileWrapper
```

Return Value

A file wrapper containing the pasteboard data, or `nil` if the receiver contained no data of type `NSFileContentsPboardType`.

Discussion

In Mac OS X v10.5 and earlier, the file contents pboard type allowed you to synthesize a pboard type for a file's contents based on the file's extension. In Mac OS X v10.5 and later, using the UTI of a file to represent its contents now replaces this functionality.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPasteboard.h

readObjectsForClasses:options:

Reads from the receiver objects that best match the specified array of classes.

```
- (NSArray *)readObjectsForClasses:(NSArray *)classArray
    options:(NSDictionary *)options
```

Parameters

classArray

An array of class objects.

The classes should appear in the array in the order the preferred order of representation. Classes in the array must conform to the NSPasteboardReading Protocol Reference protocol.

options

A dictionary that specifies options to refine the search for pasteboard items, for example to restrict the search to file URLs with particular content types. For valid dictionary keys, see "Pasteboard Reading Options" (page 1910).

Return Value
An array containing the best match (if any) for each of the items on the receiver that can be represented by a class specified in *classArray*. Returns nil if there is an error in retrieving the requested items from the pasteboard, or if no objects of the specified classes can be created.

Discussion
Classes in *classArray* must implement the NSPasteboardReading protocol. Cocoa classes that implement this protocol include NSImage, NSString, NSURL, NSColor, NSAttributedString, and NSPasteboardItem. For every item on the pasteboard, each class in the provided array will be queried for the types it can read using readableTypesForPasteboard: (page 3770). An instance is created of the first class found in the provided array whose readable types match a conforming type contained in that pasteboard item. Any instances that could be created from pasteboard item data is returned to the caller. Additional options, such as restricting the search to file URLs with particular content types, can be specified with an options dictionary.

Only objects of the requested classes are returned. You can always ensure to receive one object per item on the pasteboard by including the NSPasteboardItem class in the array of classes.

Consider the following example: there are five items on the pasteboard, two contain TIFF data, two contain RTF data, one contains a private data type. The following table shows what objects you get back in the returned array for different classes in *classArray*

Classes in *classArray*	Returned objects
NSImage	Two NSImage objects.
NSAttributedString	Two NSAttributedString objects.
NSImage, NSAttributedString	Two NSImage objects, and two NSAttributedString objects.

Classes in *classArray*	Returned objects
`NSImage`, `NSAttributedString`, `NSPasteboardItem`	Two `NSImage` objects, two `NSAttributedString` objects, and one `NSPasteboardItem` object containing the private data type.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `canReadObjectForClasses:options:` (page 1890)
- `canReadItemWithDataConformingToTypes:` (page 1890)
- `pasteboardItems` (page 1895)

Related Sample Code
DemoMonkey
LightTable

Declared In
`NSPasteboard.h`

releaseGlobally

Releases the receiver's resources in the pasteboard server.

```
- (void)releaseGlobally
```

Discussion
After this method is invoked, no other application can use the receiver.

Special Considerations

A temporary, privately named pasteboard can be released this way when it is no longer needed, but a standard pasteboard should never be released globally.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPasteboard.h`

setData:forType:

Sets the given data as the representation for the specified type for the first item on the receiver.

```
- (BOOL)setData:(NSData *)data forType:(NSString *)dataType
```

Parameters

data

The data to write to the pasteboard.

dataType
The type of data in the *data* parameter. The type must have been declared by a previous `declareTypes:owner:` (page 1893) message.

Return Value
`YES` if the data was written successfully, otherwise `NO` if ownership of the pasteboard has changed. Any other error raises an `NSPasteboardCommunicationException`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `dataForType:` (page 1892)
- `setPropertyList:forType:` (page 1899)
- `setString:forType:` (page 1900)

Related Sample Code
CocoaDragAndDrop
GLUT
PDFKitLinker2
Sketch+Accessibility
Sketch-112

Declared In
`NSPasteboard.h`

setPropertyList:forType:

Sets the given property list as the representation for the specified type for the first item on the receiver.

```
- (BOOL)setPropertyList:(id)propertyList forType:(NSString *)dataType
```

Parameters
propertyList
The property list data to write to the pasteboard.

dataType
The type of property-list data in the *propertyList* parameter. The type must have been declared by a previous `declareTypes:owner:` (page 1893) message.

Return Value
`YES` if the data was written successfully, otherwise `NO` if ownership of the pasteboard has changed. Any other error raises an `NSPasteboardCommunicationException`.

Discussion
This method invokes `setData:forType:` (page 1898) with a serialized property list parameter.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `propertyListForType:` (page 1895)
- `setString:forType:` (page 1900)

Related Sample Code
DragNDropOutlineView
FunHouse
iSpend
Quartz Composer WWDC 2005 TextEdit
ZipBrowser

Declared In
NSPasteboard.h

setString:forType:

Sets the given string as the representation for the specified type for the first item on the receiver.

```
- (BOOL)setString:(NSString *)string forType:(NSString *)dataType
```

Parameters

string
The string to write to the pasteboard.

dataType
The type of string data. The type must have been declared by a previous `declareTypes:owner:` (page 1893) message.

Return Value
`YES` if the data was written successfully, otherwise `NO` if ownership of the pasteboard has changed. Any other error raises an `NSPasteboardCommunicationException`.

Discussion
This method invokes `setData:forType:` (page 1898) to perform the write.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `stringForType:` (page 1901)
- `setData:forType:` (page 1898)
- `setPropertyList:forType:` (page 1899)

Related Sample Code
DemoAssistant
DragNDropOutlineView
QTMetadataEditor
SpotlightFortunes
With and Without Bindings

Declared In
NSPasteboard.h

stringForType:

Returns a concatenation of the strings for the specified type from all the items in the receiver that contain the type.

```
- (NSString *)stringForType:(NSString *)dataType
```

Parameters

dataType

The pasteboard data type to read.

Return Value

A concatenation of the strings for the specified type from all the items in the receiver that contain the type, or `nil` if none of the items contain strings of the specified type.

Discussion

This method invokes `dataForType:` to obtain the string. If the string cannot be obtained, `stringForType:` returns `nil`. See `dataForType:` (page 1892) for a description of what will cause `nil` to be returned.

In Mac OS X v10.6 and later, if the receiver contains multiple items that can provide string, RTF, or RTFD data, the text data from each item is returned as a combined result separated by newlines.

Special Considerations

You must send `types` (page 1901) or `availableTypeFromArray:` (page 1889) before invoking `stringForType:`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setString:forType:` (page 1900)

Related Sample Code

iSpend

SpotlightFortunes

Declared In

`NSPasteboard.h`

types

Returns an array of the receiver's supported data types.

```
- (NSArray *)types
```

Return Value

An array of `NSString` objects containing the union of the types of data declared for all the pasteboard items on the receiver. The returned types are listed in the order they were declared.

Discussion

You must send a `types` or `availableTypeFromArray:` (page 1889) message before reading any data from an `NSPasteboard` object. If you need to see if a type in the returned array matches a type string you have stored locally, use the `isEqualToString:` method to perform the comparison.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `declareTypes:owner:` (page 1893)
- `dataForType:` (page 1892)

Related Sample Code
iSpend
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSPasteboard.h`

writeFileContents:

Writes the contents of the specified file to the pasteboard.

```
- (BOOL)writeFileContents:(NSString *)filename
```

Parameters
filename
The name of the file to write to the pasteboard.

Return Value
`YES` if the data was successfully written, otherwise `NO`.

Discussion
Writes the contents of the file *filename* to the receiver and declares the data to be of type `NSFileContentsPboardType` and also of a type appropriate for the file's extension (as returned by the `NSCreateFileContentsPboardType` function when passed the files extension), if it has one.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `readFileContentsType:toFile:` (page 1896)

Declared In
`NSPasteboard.h`

writeFileWrapper:

Writes the serialized contents of the specified file wrapper to the pasteboard.

```
- (BOOL)writeFileWrapper:(NSFileWrapper *)wrapper
```

Parameters
wrapper
The file wrapper to write to the pasteboard.

Return Value
`YES` if the data was successfully written, otherwise `NO`.

Discussion
Writes the serialized contents of the file wrapper *wrapper* to the receiver and declares the data to be of type `NSFileContentsPboardType` and also of a type appropriate for the file's extension (as returned by the `NSCreateFileContentsPboardType` function when passed the files extension), if it has one. If *wrapper* does not have a preferred filename, this method raises an exception.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPasteboard.h`

writeObjects:

Writes an array of objects to the receiver.

```
- (BOOL)writeObjects:(NSArray *)objects
```

Parameters

objects

An array of objects that implement the `NSPasteboardWriting Protocol Reference` protocol (including instances of `NSPasteboardItem`).

Return Value
`YES` if the array was successfully added, otherwise `NO`.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
DemoMonkey
IconCollection

Declared In
`NSPasteboard.h`

Constants

Pasteboard Names

The `NSPasteboard` class defines the following named pasteboards.

```
NSString *NSGeneralPboard;
NSString *NSFontPboard;
NSString *NSRulerPboard;
NSString *NSFindPboard;
NSString *NSDragPboard;
```

Constants

`NSGeneralPboard`

The pasteboard that's used for ordinary cut, copy, and paste operations.

This pasteboard holds the contents of the last selection that's been cut or copied.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSFontPboard`

The pasteboard that holds font and character information and supports Copy Font and Paste Font commands that may be implemented in a text editor.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSRulerPboard`

The pasteboard that holds information about paragraph formats in support of the Copy Ruler and Paste Ruler commands that may be implemented in a text editor.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSFindPboard`

The pasteboard that holds information about the current state of the active application's find panel.

This information permits users to enter a search string into the find panel, then switch to another application to conduct another search.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSDragPboard`

The pasteboard that stores data to be moved as the result of a drag operation.

For additional information on working with the drag pasteboard, see *Drag and Drop Programming Topics for Cocoa*.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

Declared In

`AppKit/NSPasteboard.h`

Types for Standard Data (Mac OS X 10.6 and later)

The `NSPasteboard` class uses the following constants to define UTIs for common pasteboard data types.

```
NSString *const NSPasteboardTypeString;
NSString *const NSPasteboardTypePDF;
NSString *const NSPasteboardTypeTIFF;
NSString *const NSPasteboardTypePNG;
NSString *const NSPasteboardTypeRTF;
NSString *const NSPasteboardTypeRTFD;
NSString *const NSPasteboardTypeHTML;
NSString *const NSPasteboardTypeTabularText;
NSString *const NSPasteboardTypeFont;
NSString *const NSPasteboardTypeRuler;
NSString *const NSPasteboardTypeColor;
NSString *const NSPasteboardTypeSound;
NSString *const NSPasteboardTypeMultipleTextSelection;
NSString *const NSPasteboardTypeFindPanelSearchOptions;
```

Constants

`NSPasteboardTypeString`
String data.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypePDF`
PDF data.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypeTIFF`
Tag Image File Format (TIFF) data.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypePNG`
PNG image data.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypeRTF`
Rich Text Format (RTF) data.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypeRTFD`
RTFD formatted file contents.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypeHTML`
HTML data.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypeTabularText`
An `NSString` object containing tab-separated fields of text.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypeFont`
Font and character information.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypeRuler`
Paragraph formatting information.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypeColor`
Color data (an `NSColor` object).
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypeSound`
Sound data (an `NSSound` object).
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypeMultipleTextSelection`
Multiple text selection.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

`NSPasteboardTypeFindPanelSearchOptions`
Type for the Find panel metadata property list.
Available in Mac OS X v10.6 and later.
Declared in `NSPasteboard.h`.

Types for Standard Data (Mac OS X 10.5 and earlier)

The `NSPasteboard` class uses the following common pasteboard data types.

```
NSString *NSStringPboardType;
NSString *NSFilenamesPboardType;
NSString *NSPostScriptPboardType;
NSString *NSTIFFPboardType;
NSString *NSRTFPboardType;
NSString *NSTabularTextPboardType;
NSString *NSFontPboardType;
NSString *NSRulerPboardType;
NSString *NSFileContentsPboardType;
NSString *NSColorPboardType;
NSString *NSRTFDPboardType;
NSString *NSHTMLPboardType;
NSString *NSPICTPboardType;
NSString *NSURLPboardType;
NSString *NSPDFPboardType;
NSString *NSVCardPboardType;
NSString *NSFilesPromisePboardType;
NSString *NSMultipleTextSelectionPboardType;
```

Constants

`NSColorPboardType`
`NSColor` data.

On Mac OS X v10.6 and later, use `NSPasteboardTypeColor` (page 1906) (and you read and write colors directly to and from the pasteboard).

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSFileContentsPboardType`
A representation of a file's contents.

The file contents pboard type allowed you to synthesize a pboard type for a file's contents based on the file's extension.

On Mac OS X v10.6 and later, you should use the UTI of a file to represent its contents instead.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSFilenamesPboardType`
An array of `NSString` objects designating one or more filenames.

On Mac OS X v10.6 and later, use `writeObjects:` (page 1903) to write file URLs to the pasteboard.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSFontPboardType`
Font and character information.

On Mac OS X v10.6 and later, use `NSPasteboardTypeFont` (page 1906) instead.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSHTMLPboardType`
HTML (which an `NSTextView` object can read from, but not write to).

On Mac OS X v10.6 and later, use `NSPasteboardTypeHTML` (page 1905) instead.

Available in Mac OS X v10.1 and later.

Declared in `NSPasteboard.h`.

`NSPDFPboardType`
PDF data.

On Mac OS X v10.6 and later, use `NSPasteboardTypePDF` (page 1905) instead.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSPICTPboardType`
QuickDraw picture data.

The PICT format was formally deprecated in Mac OS X v10.4 along with QuickDraw. You should not be explicitly providing or looking for PICT data on the pasteboard.

To aid in this deprecation, if PICT is the only image type on the pasteboard, as is sometimes the case when copying images from 32-bit Carbon applications, a translated image type will be automatically reported and provided by `NSPasteboard`. The translated type is added to the types array ahead of PICT so that the deprecated PICT format is not the preferred format. In addition, when an application provides image data to `NSPasteboard`, the Carbon Pasteboard Manager will automatically make a PICT translation available to 32-bit Carbon applications.

Although `NSPICTPboardType`, and its UTI equivalent `kUTTypePICT`, will appear in a pasteboard's type array retrieved from the existing `NSPasteboard` API, it may cease to be reported in future releases.

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

Declared in `NSPasteboard.h`.

`NSPostScriptPboardType`
Encapsulated PostScript (EPS) code.

On Mac OS X v10.6 and later, use `@"com.adobe.encapsulated-postscript"` instead.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSRulerPboardType`
Paragraph formatting information.

On Mac OS X v10.6 and later, use `NSPasteboardTypeRuler` (page 1906) instead.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSRTFPboardType`
Rich Text Format (RTF) data.

On Mac OS X v10.6 and later, use `NSPasteboardTypeRTF` (page 1905) instead.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSRTFDPboardType`
RTFD formatted file contents.

On Mac OS X v10.6 and later, use `NSPasteboardTypeRTFD` (page 1905) instead.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSStringPboardType`

`NSString` data. (Deprecated. On Mac OS X v10.6 and later, use `NSPasteboardTypeString` (page 1905) instead.)

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSTabularTextPboardType`

An `NSString` object containing tab-separated fields of text.

On Mac OS X v10.6 and later, use `NSPasteboardTypeTabularText` (page 1906) instead.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSTIFFPboardType`

Tag Image File Format (TIFF) data.

On Mac OS X v10.6 and later, use `NSPasteboardTypeTIFF` (page 1905) instead.

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSURLPboardType`

`NSURL` data for one file or resource.

On Mac OS X v10.6 and later, use `writeObjects:` (page 1903) to write URLs directly to the pasteboard instead.

On Mac OS X v10.5 and earlier: to write an URL to a pasteboard you use `writeToPasteboard:` (page 3112) (`NSURL`); to get an URL from a pasteboard you use `URLFromPasteboard:` (page 3111) (`NSURL`).

Available in Mac OS X v10.0 and later.

Declared in `NSPasteboard.h`.

`NSVCardPboardType`

VCard data.

On Mac OS X v10.6 and later, use `(NSString *)kUTTypeVCard` instead.

Available in Mac OS X v10.2 and later.

Declared in `NSPasteboard.h`.

`NSFilesPromisePboardType`

Promised files.

On Mac OS X v10.6 and later, use `(NSString *)kPasteboardTypeFileURLPromise` instead.

For information on promised files, see Dragging Files in *Drag and Drop Programming Topics for Cocoa*.

Available in Mac OS X v10.2 and later.

Declared in `NSPasteboard.h`.

`NSInkTextPboardType`

Ink text data.

On Mac OS X v10.6 and later, use `(NSString *)kUTTypeInkText` instead.

For information on ink text objects, see *Using Ink Services in Your Application*.

Available in Mac OS X v10.4 and later.

Declared in `NSPasteboard.h`.

`NSMultipleTextSelectionPboardType`
Multiple text selection.

On Mac OS X v10.6 and later, use `NSPasteboardTypeMultipleTextSelection` (page 1906) instead.

Available in Mac OS X v10.5 and later.

Declared in `NSPasteboard.h`.

Discussion
See also `NSSoundPboardType` (page 2478) (`NSSound`).

Version Notes
Pboard types will be deprecated in a future release. On Mac OS X 10.6 and later you should replace any use of pboard types with UTIs, including the constants described in "Types for Standard Data (Mac OS X 10.6 and later)" (page 1904).

Pasteboard Reading Options

These options can be used for both the `readObjectsForClasses:options:` (page 1897) and `canReadObjectForClasses:options:` (page 1890) methods, unless otherwise specified. The currently available options allow for customization of how URLS are read from the pasteboard.

```
NSString *NSPasteboardURLReadingFileURLsOnlyKey;
NSString *NSPasteboardURLReadingContentsConformToTypesKey;
```

Constants
`NSPasteboardURLReadingFileURLsOnlyKey`
Option for reading URLs to restrict the results to file URLs only.

The value for this key is an `NSNumber` object with a boolean value.

Available in Mac OS X v10.6 and later.

Declared in `NSPasteboard.h`.

`NSPasteboardURLReadingContentsConformToTypesKey`
Option for reading URLs to restrict the results to URLs with contents that conform to any of the provided UTI types.

If the content type of a URL cannot be determined, it will not be considered to match. The value for this key is an array of UTI type strings.

Available in Mac OS X v10.6 and later.

Declared in `NSPasteboard.h`.

NSPasteboardItem Class Reference

Inherits from	NSObject
Conforms to	NSPasteboardWriting NSPasteboardReading NSObject (NSObject)
Availability	Available in Mac OS X v10.6 and later.
Declared in	NSPasteboardItem.h
Companion guides	Pasteboard Programming Guide Drag and Drop Programming Topics for Cocoa Services Implementation Guide

Overview

`NSPasteboardItem` is a generic class to represent an item on a pasteboard.

There are three main uses for an `NSPasteboardItem` object:

1. Providing data on the pasteboard.

   You can create one or more pasteboard items, set data or data providers for types, and write to them pasteboard.

2. Customizing data already on the pasteboard.

   As a delegate or subclass, you can retrieve the pasteboard items currently on the pasteboard, read the existing types and data and set new data and data providers for types as needed.

3. Retrieving data from the pasteboard.

   You can retrieve pasteboard items from the pasteboard then read the data for types you're interested in.

A pasteboard item can be associated with a single pasteboard. When you create an item, it can be written to any pasteboard. When you pass an item to a pasteboard in `writeObjects:` (page 1903), that item becomes bound to the pasteboard it was written to. When you retrieve items from a pasteboard using `pasteboardItems` (page 1895) or `readObjectsForClasses:options:` (page 1897), the returned items are associated with the messaged pasteboard. Passing an item that is already associated with a pasteboard into `writeObjects:` causes an exception to be raised.

Pasteboard items are intended to be used during a single pasteboard interaction, not held onto and used repeatedly. A pasteboard item is only valid until the owner of the pasteboard changes.

Important: If a pasteboard item is stale because the pasteboard owner has changed, it returns `nil` or `NO` values from its methods.

Tasks

Getting Types

- `types` (page 1916)
  Returns an array of UTI strings of the data types supported by the receiver.
- `availableTypeFromArray:` (page 1913)
  Returns from a given array of types the the first type contained in the pasteboard item, according to the ordering of types.

Setting the Data Provider

- `setDataProvider:forTypes:` (page 1914)
  Sets the data provider for the specified types.

Setting Values

- `setData:forType:` (page 1914)
  Sets the value for a specified type as an `NSData` object.
- `setString:forType:` (page 1915)
  Sets the value for a specified type as a string.
- `setPropertyList:forType:` (page 1915)
  Sets the value for a specified type as a property list.

Getting Values

- `dataForType:` (page 1913)
  Returns the value for the specified type as an `NSData` object.
- `stringForType:` (page 1916)
  Returns the value for the specified type as a string.
- `propertyListForType:` (page 1913)
  Returns the value for the specified type as a property list.

Instance Methods

availableTypeFromArray:

Returns from a given array of types the the first type contained in the pasteboard item, according to the ordering of types.

```
- (NSString *)availableTypeFromArray:(NSArray *)types
```

Parameters

types

An array of strings representing UTIs, arranged in order of preference (most preferred as the 0th element in the array).

Return Value

The first (according to the sender's ordering of *types*) type in *types* contained in the pasteboard item, or nil if the receiver does not contain any types given in *types*.

Discussion

The method checks for UTI conformance of the requested types, preferring an exact match to conformance.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSPasteboardItem.h`

dataForType:

Returns the value for the specified type as an `NSData` object.

```
- (NSData *)dataForType:(NSString *)type
```

Parameters

type

A UTI type string.

Return Value

The value for the specified type as an `NSData` object.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSPasteboardItem.h`

propertyListForType:

Returns the value for the specified type as a property list.

```
- (id)propertyListForType:(NSString *)type
```

Parameters

type

A UTI type string.

Return Value

The value for the specified type as a property list.

Discussion

For more about property lists, see *Property List Programming Guide*.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSPasteboardItem.h

setData:forType:

Sets the value for a specified type as an NSData object.

```
- (BOOL)setData:(NSData *)data forType:(NSString *)type
```

Parameters

data

An NSData object containing the value for the representation specified by *type*.

type

A UTI type string.

Return Value

YES if the value was set successfully, otherwise NO.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSPasteboardItem.h

setDataProvider:forTypes:

Sets the data provider for the specified types.

```
- (BOOL)setDataProvider:(id <NSPasteboardItemDataProvider>)dataProvider
    forTypes:(NSArray *)types
```

Parameters

dataProvider

A pasteboard data provider.

types

An array of strings indicating the UTIs for the data representations *dataProvider* may provide.

Return Value

YES if the data provider was set successfully, otherwise NO.

Discussion
This method registers the data provider to be messaged to provide the data for any of the specified types when requested.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSPasteboardItem.h`

setPropertyList:forType:

Sets the value for a specified type as a property list.

```
- (BOOL)setPropertyList:(id)propertyList forType:(NSString *)type
```

Parameters

propertyList
A property list object containing the value for the representation specified by *type*.

For about property lists, see *Property List Programming Guide*.

type
A UTI type string.

Return Value
`YES` if the value was set successfully, otherwise `NO`.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSPasteboardItem.h`

setString:forType:

Sets the value for a specified type as a string.

```
- (BOOL)setString:(NSString *)string forType:(NSString *)type
```

Parameters

string
A string for the representation specified by *type*.

type
A UTI type string.

Return Value
`YES` if the value was set successfully, otherwise `NO`.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSPasteboardItem.h`

stringForType:

Returns the value for the specified type as a string.

```
- (NSString *)stringForType:(NSString *)type
```

Parameters

type

A UTI type string.

Return Value

The value for the specified type as a string.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSPasteboardItem.h

types

Returns an array of UTI strings of the data types supported by the receiver.

```
- (NSArray *)types
```

Return Value

An array of UTI strings of the data types supported by the receiver.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSPasteboardItem.h

NSPathCell Class Reference

Inherits from	NSActionCell : NSCell : NSObject
Conforms to	NSOpenSavePanelDelegate NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSPathCell.h

Overview

`NSPathCell` implements the user interface of an `NSPathControl` object. It maintains a collection of `NSPathComponentCell` objects that represent a particular path to be displayed to the user.

The path shown can be set with the `setURL:` (page 1929) method. Doing so removes all displayed `NSPathComponentCell` objects and automatically fills the control with `NSPathComponentCell` objects set to have the appropriate icons, display titles, and `NSURL` values for the particular path component they represent. Alternatively, you can fill the control manually by setting the cell array or directly modifying existing cells.

Both an action and double-click action can be set for the path control. To find out what path component cell was clicked in the action, you can read the value of `clickedPathComponentCell` (page 1921). When the style is set to `NSPathStylePopUp` (page 1930), the action is still sent, and the `clickedPathComponentCell` (page 1921) value for the represented menu item is correctly set. The `clickedPathComponentCell` (page 1921) value is valid only when the action is being sent. It is also valid when the keyboard is used to invoke the action.

Automatic animated expansion of partially hidden `NSPathComponentCell` objects happens if you correctly call `mouseEntered:` (page 2165) and `mouseExited:` (page 2165) for each `NSPathComponentCell` in the `NSPathCell` object. This is not required if the `pathStyle` (page 1923) is set to `NSPathStylePopUp` (page 1930), or if you wish to not have the animation.

`NSPathCell` supports several path display styles. `NSPathStyleStandard` (page 1930) has a light blue background with arrows indicating the path. `NSPathStyleNavigationBar` (page 1930) has more defined arrows (chevrons) and looks a little like a segmented button. `NSPathStylePopUp` (page 1930) looks and works like an `NSPopUpButton` object to display the full path, or, if the cell is editable, select a new path.

If the cell's `isEditable` (page 567) method returns `YES` (the default), you can drag and drop into the cell to change the value. You can constrain what can be dropped using UTIs (Uniform Type Identifiers) with `setAllowedTypes:` (page 1925) or the appropriate delegate methods on `NSPathControl`.

If the cell's `isSelectable` (page 569) method returns `YES` (the default), the cell's contents can automatically be dragged out. The proper UTI, filename, and URL are placed on the pasteboard. You can further control or limit this by using the appropriate delegate methods on `NSPathControl`.

If the cell is editable and has the path style set to `NSPathStylePopUp` (page 1930), an additional item in the pop-up menu allows selecting another location. By default, an `NSOpenPanel` object is configured based on the allowed types. The `NSOpenPanel` object can be customized with a delegate method.

Tasks

Displaying Hidden Components

- `mouseEntered:withFrame:inView:` (page 1922)
  Displays the cell component over which the mouse is hovering.
- `mouseExited:withFrame:inView:` (page 1922)
  Hides the cell component over which the mouse is hovering.

Setting the Allowed Types

- `allowedTypes` (page 1920)
  Returns the component types allowed in the path when the cell is editable.
- `setAllowedTypes:` (page 1925)
  Sets the component types allowed in the path when the cell is editable.

Setting the Control Style

- `pathStyle` (page 1923)
  Returns the receiver's path style.
- `setPathStyle:` (page 1928)
  Sets the receiver's path style.
- `setControlSize:` (page 1926)
  Sets the receiver's control size.

Setting the Object Value

- `setObjectValue:` (page 1927)
  Sets the receiver's object value.

Setting Cell Appearance

- `placeholderAttributedString` (page 1924)
  Returns the placeholder attributed string.
- `setPlaceholderAttributedString:` (page 1928)
  Sets the value of the placeholder attributed string.
- `setPlaceholderString:` (page 1929)
  Sets the value of the placeholder string.
- `placeholderString` (page 1924)
  Returns the placeholder string.
- `setBackgroundColor:` (page 1926)
  Sets the receiver's background color.
- `backgroundColor` (page 1921)
  Returns the current background color of the receiver.

Managing Path Components

+ `pathComponentCellClass` (page 1920)
  Returns the class used to create `pathComponentCell` objects when automatically filling up the control.
- `rectOfPathComponentCell:withFrame:inView:` (page 1925)
  Returns the current rectangle being displayed for a given path component cell, with respect to a given frame in a given view.
- `pathComponentCellAtPoint:withFrame:inView:` (page 1923)
  Returns the cell located at the given point within the given frame of the given view.
- `clickedPathComponentCell` (page 1921)
  Returns the clicked cell.
- `pathComponentCells` (page 1923)
  Returns an array of the `NSPathComponentCell` objects currently being displayed.
- `setPathComponentCells:` (page 1927)
  Sets the array of `NSPathComponentCell` objects currently being displayed.

Setting the Double-Click Action

- `doubleAction` (page 1922)
  Returns the receiver's double-click action method.
- `setDoubleAction:` (page 1927)
  Sets the receiver's double-click action.

Setting the Path

- `URL` (page 1929)
  Returns the path displayed by the receiver.

- `setURL:` (page 1929)
  Sets the value of the path displayed by the receiver.

Setting the Delegate

- `delegate` (page 1921)
  Returns the receiver's delegate.
- `setDelegate:` (page 1926)
  Sets the receiver's delegate.

Class Methods

pathComponentCellClass

Returns the class used to create `pathComponentCell` objects when automatically filling up the control.

```
+ (Class)pathComponentCellClass;
```

Return Value
The class used to create `NSPathComponentCell` objects.

Discussion
Subclasses can override this method to return a custom cell class that is automatically used. By default, the method returns `[NSPathComponentCell class]`, or a specialized subclass thereof.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPathCell.h`

Instance Methods

allowedTypes

Returns the component types allowed in the path when the cell is editable.

```
- (NSArray *)allowedTypes;
```

Return Value
An array of strings representing either file extensions or UTIs. Can be `nil`, the default value, allowing all types, or an empty array, allowing nothing.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

backgroundColor

Returns the current background color of the receiver.

```
- (NSColor *)backgroundColor
```

Return Value
The background color.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

clickedPathComponentCell

Returns the clicked cell.

```
- (NSPathComponentCell *)clickedPathComponentCell
```

Return Value
The component cell that was clicked, or `nil`, if no cell has been clicked.

Discussion
The value returned is generally valid only when the action or double-click action is being sent.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

doubleAction

Returns the receiver's double-click action method.

```
- (SEL)doubleAction
```

Return Value
The action method invoked when the user double-clicks the receiver.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPathCell.h`

mouseEntered:withFrame:inView:

Displays the cell component over which the mouse is hovering.

```
- (void)mouseEntered:(NSEvent *)event withFrame:(NSRect)frame inView:(NSView *)view;
```

Parameters

event
The mouse-entered event.

frame
The frame in which the cell is located.

view
The view in which the cell is located.

Discussion
The `NSPathCell` object dynamically animates to display the component that the mouse is hovering over using mouse-entered and mouse-exited events. The control should call these methods to correctly display the hovered component to the user. The control can acquire rectangles to track using `rectOfPathComponentCell:withFrame:inView:` (page 1925).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPathCell.h`

mouseExited:withFrame:inView:

Hides the cell component over which the mouse is hovering.

```
- (void)mouseExited:(NSEvent *)event withFrame:(NSRect)frame inView:(NSView *)view;
```

Parameters

event
The mouse-exited event.

frame
The frame in which the cell is located.

view
The view in which the cell is located.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

pathComponentCellAtPoint:withFrame:inView:

Returns the cell located at the given point within the given frame of the given view.

```
- (NSPathComponentCell *)pathComponentCellAtPoint:(NSPoint)point
    withFrame:(NSRect)frame inView:(NSView *)view;
```

Parameters
point
The point within the returned cell.

frame
The frame within which the point is located.

view
The view within which the frame is located.

Return Value
The component cell within which the given point is located, or `nil` if no cell exists at that location.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

pathComponentCells

Returns an array of the `NSPathComponentCell` objects currently being displayed.

```
- (NSArray *)pathComponentCells
```

Return Value
The array of `NSPathComponentCell` objects.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

pathStyle

Returns the receiver's path style.

```
- (NSPathStyle)pathStyle
```

Return Value
The style of the path.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setPathStyle:`

Declared In
`NSPathCell.h`

placeholderAttributedString

Returns the placeholder attributed string.

```
- (NSAttributedString *)placeholderAttributedString;
```

Return Value
The placeholder attributed string.

Discussion
If the `NSPathCell` object contains no `NSPathComponentCell` objects, the placeholder attributed string is drawn in their place, if it is not `nil`. If the placeholder attributed string is `nil`, the (non-attributed) placeholder string is drawn with default attributes, if it is not `nil`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPathCell.h`

placeholderString

Returns the placeholder string.

```
- (NSString *)placeholderString;
```

Return Value
The placeholder string.

Discussion
If the `NSPathCell` object contains no `NSPathComponentCell` objects, the placeholder attributed string is drawn in their place, if it is not `nil`. If the placeholder attributed string is `nil`, the (non-attributed) placeholder string is drawn with default attributes, if it is not `nil`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPathCell.h`

rectOfPathComponentCell:withFrame:inView:

Returns the current rectangle being displayed for a given path component cell, with respect to a given frame in a given view.

```
- (NSRect)rectOfPathComponentCell:(NSPathComponentCell *)cell withFrame:(NSRect)frame
    inView:(NSView *)view;
```

Parameters

cell

The path component cell.

frame

The frame of the view in which the cell appears.

view

The view in which the cell appears.

Return Value

The rectangle occupied by the path component cell. `NSZeroRect` is returned if *cell* is not found or is not currently visible.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSPathCell.h`

setAllowedTypes:

Sets the component types allowed in the path when the cell is editable.

```
- (void)setAllowedTypes:(NSArray *)allowedTypes;
```

Parameters

allowedTypes

An array of strings representing either file extensions or UTIs. Can be `nil`, the default value, allowing all types.

Discussion

The *allowedTypes* array can contain file extensions (without the period that begins the extension) or UTIs. To allow folders, include the `public.folder` identifier. To allow any type, use `nil`. If the value of *allowedTypes* is an empty array, nothing is allowed. The default value is `nil`, allowing all types.

If the cell is editable and its type is `NSPathStylePopUp`, a Choose item is included to enable selection of a different path by invoking an Open panel. The allowed types are passed to the Open panel to filter out other types. The allowed types are also used with drag and drop to indicate if a drop is allowed.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSPathCell.h`

setBackgroundColor:

Sets the receiver's background color.

```
- (void)setBackgroundColor:(NSColor *)color
```

Parameters

color

The color to be drawn.

Discussion

By default, the background is set to a light blue color for `NSPathStyleStandard`, and `nil` for the other styles. You can use `[NSColor clearColor]` to make the background transparent.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSPathCell.h`

setControlSize:

Sets the receiver's control size.

```
- (void)setControlSize:(NSControlSize)size
```

Parameters

size

The new control size.

Discussion

`NSPathCell` properly respects the control size for the `NSPathStyleStandard` (page 1930) and `NSPathStylePopUp` (page 1930) styles. When the control size is set, the new size is propagated to subcells. When the path style is set to `NSPathStyleNavigationBar` (page 1930), you cannot change the control size, and it is always set to `NSSmallControlSize` (page 620). Attempting to change the control size when the path style is `NSPathStyleNavigationBar` (page 1930) causes an assertion. Setting the path style to `NSPathStyleNavigationBar` (page 1930) forces the control size to be `NSSmallControlSize` (page 620).

Availability

Available in Mac OS X v10.5 and later.

See Also

`- setPathStyle:` (page 1928)

`- pathStyle` (page 1923)

Declared In

`NSPathCell.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)delegate
```

Parameters

delegate

the object to set as the receiver's delegate.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

setDoubleAction:

Sets the receiver's double-click action.

```
- (void)setDoubleAction:(SEL)action
```

Parameters

action

The action method to invoke when the receiver is double-clicked.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

setObjectValue:

Sets the receiver's object value.

```
- (void)setObjectValue:(id <NSCopying>)obj;
```

Parameters

obj

The new object value for the cell.

Discussion
If `setObjectValue:` is called with an `NSURL` object, `setURL:` (page 1929) is automatically called. The `objectValue` (page 572) method returns the most recently set URL value. The `setObjectValue:` method can also take a string value, with the items separated by the path separator (`/`). Any other value is a programming error and will cause an assertion.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

setPathComponentCells:

Sets the array of `NSPathComponentCell` objects currently being displayed.

```
- (void)setPathComponentCells:(NSArray *)cells
```

Parameters

cells

An array of `NSPathComponentCell` objects.

Discussion

Each item in the array must be an instance of `NSPathComponentCell` or a subclass thereof. You cannot set this value to `nil`, but you can set it to an empty array using, for example, `[NSArray array]`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSPathCell.h`

setPathStyle:

Sets the receiver's path style.

```
- (void)setPathStyle:(NSPathStyle)style
```

Parameters

style

The new path style.

Discussion

See `setControlSize:` (page 1926) for information about path style and control size dependencies.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `pathStyle` (page 1923)
- `setControlSize:` (page 1926)

Declared In

`NSPathCell.h`

setPlaceholderAttributedString:

Sets the value of the placeholder attributed string.

```
- (void)setPlaceholderAttributedString:(NSAttributedString *)string;
```

Parameters

string

The string to set for the placeholder attributed string.

Discussion

If the `NSPathCell` object contains no `NSPathComponentCell` objects, the placeholder attributed string is drawn in their place, if it is not `nil`. If the placeholder attributed string is `nil`, the (non-attributed) placeholder string is drawn with default attributes, if it is not `nil`.

Availability

Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

setPlaceholderString:

Sets the value of the placeholder string.

```
- (void)setPlaceholderString:(NSString *)string;
```

Parameters

string
The string to set for the placeholder.

Discussion
If the NSPathCell object contains no NSPathComponentCell objects, the placeholder attributed string is drawn in their place, if it is not nil. If the placeholder attributed string is nil, the (non-attributed) placeholder string is drawn with default attributes, if it is not nil.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

setURL:

Sets the value of the path displayed by the receiver.

```
- (void)setURL:(NSURL *)url
```

Parameters

url
The new path value to display.

Discussion
When setting, an array of NSPathComponentCell objects is automatically set, based on the path in *url*. The type of NSPathComponentCell objects created can be controlled by subclassing NSPathCell and overriding pathComponentCellClass (page 1920).

If *url* is a file URL (returns YES from isFileURL), the images are automatically filled with file icons, if the path exists. The URL value itself is stored in the objectValue property of the cell.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathCell.h

URL

Returns the path displayed by the receiver.

```
- (NSURL *)URL
```

Return Value
The path value.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPathCell.h`

Constants

NSPathStyle

`NSPathStyle` constants represent the different visual and behavioral styles an `NSPathControl` or `NSPathCell` object can have.

```
enum {
    NSPathStyleStandard,
    NSPathStyleNavigationBar,
    NSPathStylePopUp,
};
typedef NSInteger NSPathStyle;
```

Constants

`NSPathStyleStandard`
The standard display style and behavior. All path component cells are displayed with an icon image and component name. If the path can not fully be displayed, the middle parts are truncated as required.

Available in Mac OS X v10.5 and later.

Declared in `NSPathCell.h`.

`NSPathStyleNavigationBar`
The navigation bar display style and behavior. Similar to the `NSPathStyleStandard` with the navigation bar drawing style. Also known as the breadcrumb style.

Available in Mac OS X v10.5 and later.

Declared in `NSPathCell.h`.

`NSPathStylePopUp`
The pop-up display style and behavior. Only the last path component is displayed with an icon image and component name. The full path is shown when the user clicks on the cell. If the cell is editable, a Choose item is included to enable selecting a different path.

Available in Mac OS X v10.5 and later.

Declared in `NSPathCell.h`.

Declared In
`NSPathCell.h`

NSPathComponentCell Class Reference

Inherits from	NSTextFieldCell : NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSPathComponentCell.h
Related sample code	DesktopImage ObjectPath

Overview

The `NSPathComponentCell` class displays a component of a path.

An `NSPathCell` object manages a collection of `NSPathComponentCell` objects, in conjunction with an `NSPathControl` object, to represent a path.

Tasks

Setting the Image

- `image` (page 1932)
  Returns the image displayed for this component cell.
- `setImage:` (page 1932)
  Sets the image displayed for this component cell.

Setting the Path

- `URL` (page 1933)
  Returns the portion of the path from the root through the component represented by the receiver.

- setURL: (page 1933)
  Sets the value of the portion of the path from the root through the component represented by the receiver.

Instance Methods

image

Returns the image displayed for this component cell.

```
- (NSImage *)image;
```

Return Value
The component cell image.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setImage (page 1932)

Declared In
NSPathComponentCell.h

setImage:

Sets the image displayed for this component cell.

```
- (void)setImage:(NSImage *)value;
```

Parameters
value
The image to set for this component cell.

Discussion
Generally, a 16-by-16–point image fits best when the path style is NSPathStyleStandard or NSPathStylePopUp, and a 14-by-14–point image is best when the path style is NSPathStyleNavigationBar.

Availability
Available in Mac OS X v10.5 and later.

See Also
- image (page 1932)

Related Sample Code
ObjectPath

Declared In
NSPathComponentCell.h

setURL:

Sets the value of the portion of the path from the root through the component represented by the receiver.

```
- (void)setURL:(NSURL *)url
```

Parameters

url

The new path value to display.

Availability
Available in Mac OS X v10.5 and later.

See Also
- URL (page 1933)

Related Sample Code
ObjectPath

Declared In
NSPathComponentCell.h

URL

Returns the portion of the path from the root through the component represented by the receiver.

```
- (NSURL *)URL
```

Return Value
The path value.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setURL: (page 1933)

Related Sample Code
DesktopImage
ObjectPath

Declared In
NSPathComponentCell.h

NSPathControl Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSPathControl.h
Related sample code	DesktopImage ObjectPath PhotoSearch

Overview

`NSPathControl` is a subclass of `NSControl` that represents a file system path or virtual path.

The `NSPathControl` class uses `NSPathCell` to implement its user interface. `NSPathControl` provides cover methods for most `NSPathCell` methods—the cover method simply invokes the corresponding cell method. See also `NSPathComponentCell`, which represents individual components of the path, and two associated protocols: `NSPathCellDelegate` and `NSPathControlDelegate`.

`NSPathControl` has three styles represented by the `NSPathStyle` (page 1930) enumeration constants `NSPathStyleStandard` (page 1930), `NSPathStyleNavigationBar` (page 1930), and `NSPathStylePopUp` (page 1930). The represented path can be a file system path or any other type of path leading through a sequence of nodes or components, as defined by the programmer.

`NSPathControl` automatically supports drag and drop, which can be further customized via delegate methods. To accept drag and drop, `NSPathControl` calls `registerForDraggedTypes:` (page 3201) with `NSFilenamesPboardType` (page 1907) and `NSURLPboardType` (page 1909). When the URL value in the `NSPathControl` object changes because of an automatic drag and drop operation or the user selecting a new path via the open panel, the action is sent. On Mac OS X v10.5 the value returned by `clickedPathComponentCell` (page 1937) is `nil`, on Mac OS X v10.6 and later, `clickedPathComponentCell` (page 1937) returns the clicked cell.

Tasks

Setting the Control Style

- `pathStyle` (page 1939)
  Returns the receiver's path style.
- `setPathStyle:` (page 1942)
  Sets the receiver's path style.

Setting the Background Color

- `setBackgroundColor:` (page 1940)
  Sets the receiver's background color.
- `backgroundColor` (page 1937)
  Returns the current background color of the receiver.

Managing Path Components

- `clickedPathComponentCell` (page 1937)
  Returns the clicked cell.
- `pathComponentCells` (page 1939)
  Returns an array of the `NSPathComponentCell` objects currently being displayed.
- `setPathComponentCells:` (page 1942)
  Sets the array of `NSPathComponentCell` objects currently being displayed.

Setting the Double-Click Action

- `doubleAction` (page 1938)
  Returns the receiver's double-click action method.
- `setDoubleAction:` (page 1941)
  Sets the receiver's double-click action.

Setting the Path

- `URL` (page 1943)
  Returns the path value displayed by the receiver.
- `setURL:` (page 1943)
  Sets the path value displayed by the receiver.

Setting the Delegate

- `delegate` (page 1938)
  Returns the receiver's delegate.
- `setDelegate:` (page 1940)
  Sets the receiver's delegate.

Setting the Drag Operation Mask

- `setDraggingSourceOperationMask:forLocal:` (page 1941)
  Configures the default value returned from `draggingSourceOperationMaskForLocal:` (page 3670).

Setting Popup Menu

- `menu` (page 1939)
  Returns the menu that is used for the path control's cells.
- `setMenu:` (page 1941)
  Sets the menu used for the path control's cells.

Instance Methods

backgroundColor

Returns the current background color of the receiver.

```
- (NSColor *)backgroundColor
```

Return Value
The background color.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setBackgroundColor:` (page 1940)

Declared In
`NSPathControl.h`

clickedPathComponentCell

Returns the clicked cell.

```
- (NSPathComponentCell *)clickedPathComponentCell
```

Return Value
The component cell that was clicked.

Discussion
The value returned is generally valid only when the action or double action is being sent.

> **Note:** In Mac OS X 10.5 and earlier the returned value was [nil] if no cell had been clicked. In Mac OS X 10.6, the folder of the cell that the user selected is returned instead.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `pathComponentCells` (page 1939)

Related Sample Code
DesktopImage

Declared In
`NSPathControl.h`

delegate

Returns the receiver's delegate.

```
- (id < NSPathControlDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setDelegate:` (page 1940)

Declared In
`NSPathControl.h`

doubleAction

Returns the receiver's double-click action method.

```
- (SEL)doubleAction
```

Return Value
The action method invoked when the user double-clicks the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setDoubleAction: (page 1941)

Declared In
NSPathControl.h

menu

Returns the menu that is used for the path control's cells.

```
- (NSMenu *)menu
```

Return Value
An instance of NSMenu.

Discussion
This method overrides the NSView implementation of menu and forwards the message to the NSPathControlCell.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSPathControl.h

pathComponentCells

Returns an array of the NSPathComponentCell objects currently being displayed.

```
- (NSArray *)pathComponentCells
```

Return Value
The array of NSPathComponentCell objects.

Availability
Available in Mac OS X v10.5 and later.

See Also
- clickedPathComponentCell (page 1937)
- setPathComponentCells: (page 1942)

Declared In
NSPathControl.h

pathStyle

Returns the receiver's path style.

```
- (NSPathStyle)pathStyle
```

Return Value
The style of the path control.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setPathStyle:` (page 1942)

Declared In
`NSPathControl.h`

setBackgroundColor:

Sets the receiver's background color.

```
- (void)setBackgroundColor:(NSColor *)color
```

Parameters

color
The color to draw.

Discussion
By default, the background is set to a light blue color for `NSPathStyleStandard` and `nil` for the other styles. You can use `[NSColor clearColor]` to make the background transparent.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `backgroundColor` (page 1937)

Related Sample Code
ObjectPath

Declared In
`NSPathControl.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSPathControlDelegate >)delegate
```

Parameters

delegate
The object to set as the receiver's delegate.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `delegate` (page 1938)

Declared In
`NSPathControl.h`

setDoubleAction:

Sets the receiver's double-click action.

```
- (void)setDoubleAction:(SEL)action
```

Parameters

action

The action method to invoke when the receiver is double-clicked.

Availability
Available in Mac OS X v10.5 and later.

See Also
- doubleAction (page 1938)

Declared In
NSPathControl.h

setDraggingSourceOperationMask:forLocal:

Configures the default value returned from draggingSourceOperationMaskForLocal: (page 3670).

```
- (void)setDraggingSourceOperationMask:(NSDragOperation)mask forLocal:(BOOL)isLocal
```

Parameters

mask

The types of drag operations allowed.

isLocal

If YES, *mask* applies when the drag destination object is in the same application as the receiver; if NO, *mask* applies when the destination object is outside the receiver's application.

Discussion
By default, draggingSourceOperationMaskForLocal: (page 3670) returns NSDragOperationEvery when *isLocal* is YES and NSDragOperationNone when *isLocal* is NO.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPathControl.h

setMenu:

Sets the menu used for the path control's cells.

```
- (void)setMenu:(NSMenu *)menu
```

Parameters

menu

An instance of NSMenu.

Discussion
This method overrides the `NSView` implementation of `setMenu:` and forwards the message to the `NSPathControlCell`.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSPathControl.h`

setPathComponentCells:

Sets the array of `NSPathComponentCell` objects currently being displayed.

```
- (void)setPathComponentCells:(NSArray *)cells
```

Parameters

cells
An array of `NSPathComponentCell` objects.

Discussion
Each item in the array must be an instance of `NSPathComponentCell` or a subclass thereof. You cannot set this value to `nil`, but you can set it to an empty array using, for example, `[NSArray array]`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- pathComponentCells` (page 1939)

Related Sample Code
ObjectPath

Declared In
`NSPathControl.h`

setPathStyle:

Sets the receiver's path style.

```
- (void)setPathStyle:(NSPathStyle)style
```

Parameters

style
The new path style.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- pathStyle` (page 1939)

Related Sample Code
ObjectPath

Declared In
NSPathControl.h

setURL:

Sets the path value displayed by the receiver.

```
- (void)setURL:(NSURL *)url
```

Parameters

url

The new path value to display.

Discussion
When setting, an array of `NSPathComponentCell` objects is automatically set based on the path in *url*. If *url* is a file URL (returns `YES` from `isFileURL`), the images are automatically filled with file icons, if the path exists. The URL value itself is stored in the `objectValue` property of the cell.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `URL` (page 1943)

Related Sample Code
DesktopImage

Declared In
NSPathControl.h

URL

Returns the path value displayed by the receiver.

```
- (NSURL *)URL
```

Return Value
The path value.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setURL:` (page 1943)

Declared In
NSPathControl.h

NSPDFImageRep Class Reference

Inherits from	NSImageRep : NSObject
Conforms to	NSCoding (NSImageRep) NSCopying (NSImageRep) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSPDFImageRep.h
Companion guide	Cocoa Drawing Guide
Related sample code	PDFView

Overview

An `NSPDFImageRep` object can render an image from a PDF format data stream.

Tasks

Creating an NSPDFImageRep

- `initWithData:` (page 1947) Deprecated in Mac OS X v10.6
  Returns an `NSPDFImageRep` object initialized with the specified PDF data.

+ `imageRepWithData:` (page 1946) Deprecated in Mac OS X v10.6
  Creates and returns an `NSPDFImageRep` object initialized with the specified PDF data.

Getting Image Data

- `currentPage` (page 1947)
  Gets the page currently displayed by the image representation.

- `pageCount` (page 1947)
  Returns the number of pages in the receiver.

- `setCurrentPage:` (page 1948)
  Sets the page to display to the specified value.
- `PDFRepresentation` (page 1948) Deprecated in Mac OS X v10.6
  Returns the PDF representation of the receiver's image.
- `bounds` (page 1946) Deprecated in Mac OS X v10.6
  Returns the receiver's bounding rectangle.

Class Methods

imageRepWithData:

Creates and returns an `NSPDFImageRep` object initialized with the specified PDF data.

```
+ (id)imageRepWithData:(NSData *)pdfData
```

Parameters

pdfData
A data object containing the PDF data for the image.

Return Value
An initialized `NSPDFImageRep` object or `nil` if the object could not be initialized. Initialization may fail if the PDF data does not conform to the PDF file format.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initWithData:` (page 1947)
- `PDFRepresentation` (page 1948)

Declared In
`NSPDFImageRep.h`

Instance Methods

bounds

Returns the receiver's bounding rectangle.

```
- (NSRect)bounds
```

Return Value
The bounding rectangle. This value is equivalent to the crop box specified by the PDF data.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
PDFView

Declared In
`NSPDFImageRep.h`

currentPage

Gets the page currently displayed by the image representation.

```
- (NSInteger)currentPage
```

Return Value
A zero-based index indicating the page being displayed.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setCurrentPage:` (page 1948)

Declared In
`NSPDFImageRep.h`

initWithData:

Returns an `NSPDFImageRep` object initialized with the specified PDF data.

```
- (id)initWithData:(NSData *)pdfData
```

Parameters
pdfData
A data object containing the PDF data for the image.

Return Value
An initialized `NSPDFImageRep` object or `nil` if the object could not be initialized. Initialization may fail if the PDF data does not conform to the PDF file format.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ imageRepWithData:` (page 1946)
`- PDFRepresentation` (page 1948)

Declared In
`NSPDFImageRep.h`

pageCount

Returns the number of pages in the receiver.

```
- (NSInteger)pageCount
```

Return Value
The number of pages in the PDF data.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
PDFView

Declared In
NSPDFImageRep.h

PDFRepresentation

Returns the PDF representation of the receiver's image.

```
- (NSData *)PDFRepresentation
```

Return Value
The PDF data used to create the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPDFImageRep.h

setCurrentPage:

Sets the page to display to the specified value.

```
- (void)setCurrentPage:(NSInteger)page
```

Parameters

page
A zero-based index indicating the page you want to display.

Availability
Available in Mac OS X v10.0 and later.

See Also
- currentPage (page 1947)

Related Sample Code
PDFView

Declared In
NSPDFImageRep.h

NSPersistentDocument Class Reference

Inherits from	NSDocument : NSObject
Conforms to	NSUserInterfaceValidations (NSDocument) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	AppKit/NSPersistentDocument.h
Companion guides	NSPersistentDocument Core Data Tutorial Document-Based Applications Overview Core Data Programming Guide
Related sample code	DerivedProperty File Wrappers with Core Data Documents LightTable Reviews TwoManyControllers

Overview

The `NSPersistentDocument` class is a subclass of `NSDocument` that is designed to easily integrate into the Core Data framework. It provides methods to access a document-wide `NSManagedObjectContext` object, and provides default implementations of methods to read and write files using the persistence framework. In a persistent document, the undo manager functionality is taken over by managed object context.

Standard document behavior is implemented as follows:

- Opening a document invokes `configurePersistentStoreCoordinatorForURL:ofType:modelConfiguration:storeOptions:error:` (page 1952) with the new URL, and adds a store of the default type (XML). Objects are loaded from the persistent store on demand through the document's context.
- Saving a new document adds a store of the default type with the chosen URL and invokes save: on the context. For an existing document, a save just invokes `save:` on the context.
- Save As for a new document, simply invokes save. For an opened document, migrates the persistent store to the new URL, and invokes `save:` on the context.
- Revert resets the document's managed object context. Objects are subsequently loaded from the persistent store on demand, as with opening a new document.

Note that `NSPersistentDocument` does not support some standard document behavior, in particular `NSPersistentDocument` does not support file wrappers. "Save To…" and Autosave are not directly supported—Core Data cannot save to a store and maintain the same changed state in the managed object context, all the while maintaining an unsaved stack as the current document.

By default an `NSPersistentDocument` instance creates its own ready-to-use persistence stack including managed object context, persistent object store coordinator and persistent store. There is a one-to-one mapping between the document and the backing object store.

You can customize the architecture of the persistence stack by overriding the methods managedObjectModel (page 1954) and `configurePersistentStoreCoordinatorForURL:ofType:modelConfiguration:storeOptions:error:` (page 1952). You might wish to do this, for example, to specify a particular managed object model.

Tasks

Managing the Persistence Objects

`- managedObjectContext` (page 1953)
Returns the managed object context for the receiver.

`- managedObjectModel` (page 1954)
Returns the receiver's managed object model.

`- setManagedObjectContext:` (page 1957)
Sets the receiver's managed object context.

`- configurePersistentStoreCoordinatorForURL:ofType:modelConfiguration:storeOptions:error:` (page 1952)
Configures the receiver's persistent store coordinator with the appropriate stores for a given URL.

`- persistentStoreTypeForFileType:` (page 1955)
Returns the type of persistent store associated with the specified file type.

Undo Support

`- hasUndoManager` (page 1953)
Returns `YES`.

`- setHasUndoManager:` (page 1956)
Overridden to be a no-op.

`- setUndoManager:` (page 1957)
Overridden to be a no-op.

`- isDocumentEdited` (page 1953)
Returns a Boolean value that indicates whether the receiver's managed object context, or editors registered with the context, have uncommitted changes.

Document Content Management

- `readFromURL:ofType:error:` (page 1955)
  Sets the contents of the receiver by reading from a file of a given type located by a given URL.
- `revertToContentsOfURL:ofType:error:` (page 1956)
  Overridden to clean up the managed object context and controllers during a revert.
- `writeToURL:ofType:forSaveOperation:originalContentsURL:error:` (page 1957)
  Saves changes in the document's managed object context and saves the document's persistent store to a given URL.

Deprecated

- `configurePersistentStoreCoordinatorForURL:ofType:error:` (page 1951) Deprecated in Mac OS X v10.5
  Configures the receiver's persistent store coordinator for a given URL and document type. (Deprecated. Use `configurePersistentStoreCoordinatorForURL:ofType:modelConfiguration:storeOptions:error:` (page 1952) instead.)

Instance Methods

configurePersistentStoreCoordinatorForURL:ofType:error:

Configures the receiver's persistent store coordinator for a given URL and document type. (Deprecated in Mac OS X v10.5. Use `configurePersistentStoreCoordinatorForURL:ofType:modelConfiguration:storeOptions:error:` (page 1952) instead.)

```
- (BOOL)configurePersistentStoreCoordinatorForURL:(NSURL *)url
    ofType:(NSString *)fileType
    error:(NSError **)error
```

Parameters

url
An URL that specifies the location of the document's store.

fileType
The document type.

error
If the method does not complete successfully, upon return contains an `NSError` object that describes the problem.

Return Value
`YES` if the method completes successfully, otherwise `NO`.

Discussion
This method is invoked automatically when an existing document is opened. You override this method to customize creation of a persistent store for a given document or store type. You can retrieve the persistent store coordinator with the following code:

```
[[self managedObjectContext] persistentStoreCoordinator];
```

Availability
Available in Mac OS X v10.4.
Deprecated in Mac OS X v10.5.

See Also
`- persistentStoreTypeForFileType:` (page 1955)
`- configurePersistentStoreCoordinatorForURL:ofType:modelConfiguration:storeOptions:error:` (page 1952)

Declared In
`NSPersistentDocument.h`

configurePersistentStoreCoordinatorForURL:ofType:modelConfiguration:storeOptions:error:

Configures the receiver's persistent store coordinator with the appropriate stores for a given URL.

```
- (BOOL)configurePersistentStoreCoordinatorForURL:(NSURL *)url
    ofType:(NSString *)fileType
    modelConfiguration:(NSString *)configuration
    storeOptions:(NSDictionary *)storeOptions
    error:(NSError **)error
```

Parameters

url
An URL that specifies the location of the document's store.

fileType
The document type.

configuration
The name of the managed object model configuration to use. (The managed object model is associated with the persistent store coordinator.) Pass `nil` if you do not want to specify a configuration.

storeOptions
Options for the store. See "Store Options" in `NSPersistentStoreCoordinator` for possible values.

error
If the method does not complete successfully, upon return contains an `NSError` object that describes the problem.

Return Value
`YES` if the method completes successfully, otherwise `NO`.

Discussion
This method is invoked automatically when an existing document is opened. You override this method to customize creation of a persistent store for a given document or store type. You can retrieve the persistent store coordinator with the following code:

```
[[self managedObjectContext] persistentStoreCoordinator];
```

You can override this method to create the store to save to or load from (invoked from within the other `NSDocument` methods to read/write files), which gives developers the ability to load/save from/to different persistent store types (default type is XML).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPersistentDocument.h`

hasUndoManager

Returns `YES`.

```
- (BOOL)hasUndoManager
```

Return Value
`YES`.

Special Considerations
You should not override this method.

See Also
`- managedObjectContext` (page 1953)

isDocumentEdited

Returns a Boolean value that indicates whether the receiver's managed object context, or editors registered with the context, have uncommitted changes.

```
- (BOOL)isDocumentEdited
```

Return Value
`YES` if the receiver's managed object context, or editors registered with the context, have uncommitted changes, otherwise `NO`.

See Also
`- managedObjectContext` (page 1953)

managedObjectContext

Returns the managed object context for the receiver.

```
- (NSManagedObjectContext *)managedObjectContext
```

Return Value
The managed object context for the receiver.

Discussion
If a managed object context for the receiver does not exist, one is created automatically. You override this method to customize the creation of the persistence stack.

Availability
Available in Mac OS X v10.4 and later.

See Also
- managedObjectModel (page 1954)

Related Sample Code
Departments and Employees
QTMetadataEditor

Declared In
NSPersistentDocument.h

managedObjectModel

Returns the receiver's managed object model.

```
- (id)managedObjectModel
```

Return Value
The receiver's managed object model, used to configure the receiver's persistent store coordinator.

Discussion
By default the Core Data framework creates a merged model from all models in the application bundle ([NSBundle mainBundle]). You can override this method to return a specific model to use to create persistent stores. A typical implementation might include code similar to the following fragment:

```
NSBundle *bundle = [NSBundle bundleForClass:[self class]];
NSString *path = [bundle pathForResource:@"MyModel" ofType:@"mom"];
NSURL *url = [NSURL fileURLWithPath:path];
NSManagedObjectModel *model = [[NSManagedObjectModel alloc]
initWithContentsOfURL:url];
```

Normally you would cache the model as an instance variable. If all your document instances use the same model, however, you can increase the efficiency of this method by caching a single instance, as illustrated in the following example.

```
- (id)managedObjectModel {
    static id sharedModel = nil;
    if (sharedModel == nil) {
        sharedModel = [[super managedObjectModel] retain];
    }
    return sharedModel;
}
```

Special Considerations

In applications built on Mac OS X v10.4, by default the Core Data framework creates a merged model from all the models found in the application bundle *and the frameworks against which the application is linked*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- managedObjectContext (page 1953)

Declared In
NSPersistentDocument.h

persistentStoreTypeForFileType:

Returns the type of persistent store associated with the specified file type.

```
- (NSString *)persistentStoreTypeForFileType:(NSString *)fileType
```

Parameters

fileType

A document file type.

Return Value

The type of persistent store associated with *fileType*. For possible values, see NSPersistentStoreCoordinator.

Discussion

You set the persistent store type in the application's property list (see Storing Document Types Information in a Property List).

Availability

Available in Mac OS X v10.4 and later.

See Also

- configurePersistentStoreCoordinatorForURL:ofType:modelConfiguration:storeOptions:error: (page 1952)

Declared In

NSPersistentDocument.h

readFromURL:ofType:error:

Sets the contents of the receiver by reading from a file of a given type located by a given URL.

```
- (BOOL)readFromURL:(NSURL *)absoluteURL
    ofType:(NSString *)typeName
    error:(NSError **)outError
```

Parameters

absoluteURL

An URL that specifies the location from which to read the document.

typeName

The document type at *absoluteURL*.

outError

If *absoluteURL* is not valid, or the store at *absoluteURL* cannot be read, upon return contains an NSError object that describes the problem

Return Value

YES if *absoluteURL* is valid and the file is read correctly, otherwise NO.

Discussion

This method sets the URL for the persistent object store associated with the receiver's managed object context to *absoluteURL*.

Availability

Available in Mac OS X v10.4 and later.

See Also
- `revertToContentsOfURL:ofType:error:` (page 1956)
- `writeToURL:ofType:forSaveOperation:originalContentsURL:error:` (page 1957)
- `configurePersistentStoreCoordinatorForURL:ofType:modelConfiguration:storeOptions:error:` (page 1952)

Declared In
`NSPersistentDocument.h`

revertToContentsOfURL:ofType:error:

Overridden to clean up the managed object context and controllers during a revert.

```
- (BOOL)revertToContentsOfURL:(NSURL *)inAbsoluteURL
    ofType:(NSString *)inTypeName
    error:(NSError **)outError
```

Parameters

inAbsoluteURL
An URL object that specifies the location of the file to which to revert.

inTypeName
The type of the document at *inAbsoluteURL*.

outError
If the method fails to complete correctly, upon return contains an `NSError` object that describes the problem.

Return Value
`YES` if the method completes correctly, otherwise `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `readFromURL:ofType:error:` (page 1955)
- `writeToURL:ofType:forSaveOperation:originalContentsURL:error:` (page 1957)

Related Sample Code
QTMetadataEditor

Declared In
`NSPersistentDocument.h`

setHasUndoManager:

Overridden to be a no-op.

```
- (void)setHasUndoManager:(BOOL)flag
```

Parameters

flag
This value is ignored.

Special Considerations

You should not override this method. The persistent document uses the managed object context's undo manager.

See Also

- managedObjectContext (page 1953)

setManagedObjectContext:

Sets the receiver's managed object context.

```
- (void)setManagedObjectContext:(NSManagedObjectContext *)managedObjectContext
```

Parameters

managedObjectContext

The managed object context for the receiver.

Availability

Available in Mac OS X v10.4 and later.

See Also

- managedObjectContext (page 1953)
- managedObjectModel (page 1954)

Declared In

NSPersistentDocument.h

setUndoManager:

Overridden to be a no-op.

```
- (void)setUndoManager:(NSUndoManager *)undoManager
```

Parameters

undoManager

This value is ignored.

Special Considerations

You should not override this method. The persistent document uses the managed object context's undo manager.

See Also

- managedObjectContext (page 1953)

writeToURL:ofType:forSaveOperation:originalContentsURL:error:

Saves changes in the document's managed object context and saves the document's persistent store to a given URL.

```
- (BOOL)writeToURL:(NSURL *)absoluteURL
    ofType:(NSString *)typeName
    forSaveOperation:(NSSaveOperationType)saveOperation
    originalContentsURL:(NSURL *)absoluteOriginalContentsURL
    error:(NSError **)error
```

Parameters

absoluteURL

An URL that specifies the new location for the document store. It must not be a relative URL.

typeName

The document type.

saveOperation

The save operation type. See the "Constants" section in `NSDocument` for possible values.

absoluteOriginalContentsURL

An URL that specifies the location of the original document store.

error

If the save fails to complete correctly, upon return contains an `NSError` object that describes the problem.

Return Value

`YES` if the save completes correctly, otherwise `NO`.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- readFromURL:ofType:error:` (page 1955)

`- revertToContentsOfURL:ofType:error:` (page 1956)

`- configurePersistentStoreCoordinatorForURL:ofType:modelConfiguration:storeOptions:error:` (page 1952)

Declared In

`NSPersistentDocument.h`

NSPICTImageRep Class Reference

Inherits from	NSImageRep : NSObject
Conforms to	NSCoding (NSImageRep) NSCopying (NSImageRep) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSPICTImageRep.h
Companion guide	Cocoa Drawing Guide

Overview

An `NSPICTImageRep` object renders an image from a PICT format data stream as described in the Carbon QuickDraw Manager documentation. This class can render PICT format version 1, version 2, and extended version 2 pictures.

Warning: There is no guarantee that the image will render exactly the same as it would under QuickDraw because of the differences between the display medium and QuickDraw. In particular, some transfer modes and region operations may not be supported.

Tasks

Creating an NSPICTImageRep

`+ imageRepWithData:` (page 1960)
Creates and returns an `NSPICTImageRep` object initialized with the specified data.

`- initWithData:` (page 1961)
Returns an `NSPICTImageRep` object initialized with the specified data.

Getting Image Data

- `boundingBox` (page 1960)
  Returns the rectangle that bounds the receiver.
- `PICTRepresentation` (page 1961)
  Returns the receiver's PICT data.

Class Methods

imageRepWithData:

Creates and returns an `NSPICTImageRep` object initialized with the specified data.

```
+ (id)imageRepWithData:(NSData *)pictData
```

Parameters

pictData
A data object containing the PICT data.

Return Value
An initialized `NSPICTImageRep` or `nil` if the object could not be initialized. Initialization may fail if the data does not conform to the PICT file format.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `PICTRepresentation` (page 1961)
- `initWithData:` (page 1961)

Declared In
`NSPICTImageRep.h`

Instance Methods

boundingBox

Returns the rectangle that bounds the receiver.

```
- (NSRect)boundingBox
```

Return Value
The rectangle bounding the receiver. This rectangle is obtained from the the *picFrame* field in the picture header. See the Carbon QuickDraw Manager documentation for information on the picture header

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPICTImageRep.h

initWithData:

Returns an NSPICTImageRep object initialized with the specified data.

```
- (id)initWithData:(NSData *)pictData
```

Parameters

pictData

A data object containing the PICT data.

Return Value
An initialized NSPICTImageRep or nil if the object could not be initialized. Initialization may fail if the data does not conform to the PICT file format.

Discussion
If the PICT data is obtained directly from a PICT file or document, this method ignores most of the 512-byte header that occurs before the start of the actual picture data. It may retrieve some relevant meta information from the header.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ imageRepWithData: (page 1960)
- PICTRepresentation (page 1961)

Declared In
NSPICTImageRep.h

PICTRepresentation

Returns the receiver's PICT data.

```
- (NSData *)PICTRepresentation
```

Return Value
A data object containing the PICT data. The returned data does not include the 512-byte header, if it was present in the original data. If you want to write the returned data to a file, you must precede it with a 512-byte header (containing all zeros) if you want to conform to the PICT document format.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPICTImageRep.h

NSPopUpButton Class Reference

Inherits from	NSButton : NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSButton) NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSPopUpButton.h
Companion guide	Application Menu and Pop-up List Programming Topics
Related sample code	ColorMatching FinalCutPro_AppleEvents MenuMadness Quartz Composer WWDC 2005 TextEdit WhackedTV

Class at a Glance

An `NSPopUpButton` object controls a pop-up menu or a pull-down menu from which a user can select an item.

Principal Attributes

- An NSMenu

Interface Builder
: Use Interface Builder to add a pop-up or pull-down menu to a window or panel.

Commonly Used Methods

`selectedItem` (page 1979)
: Returns the currently selected item.

`indexOfSelectedItem` (page 1971)
Returns an integer identifying the currently selected item.

`titleOfSelectedItem` (page 1985)
Returns a string identifying the currently selected item.

Overview

The `NSPopUpButton` class defines objects that implement the pop-up and pull-down menus of the graphical user interface.

An `NSPopUpButton` object uses an NSPopUpButtonCell object to implement its user interface.

Note that while a menu is tracking, adding, removing, or changing items on the menu is not reflected.

Tasks

Initializing an NSPopUpButton

`- initWithFrame:pullsDown:` (page 1971)
Returns an `NSPopUpButton` object initialized to the specified dimensions.

Setting the Type of Menu

`- setPullsDown:` (page 1983)
Sets whether the receiver behaves as a pull-down or pop-up menu.

`- pullsDown` (page 1977)
Returns a Boolean value indicating the behavior of the control's menu.

`- setAutoenablesItems:` (page 1982)
Sets whether the receiver automatically enables and disables its items every time a user event occurs.

`- autoenablesItems` (page 1968)
Returns whether the receiver automatically enables and disables its items every time a user event occurs.

Inserting and Deleting Items

`- addItemWithTitle:` (page 1967)
Adds an item with the specified title to the end of the menu.

`- addItemsWithTitles:` (page 1967)
Adds multiple items to the end of the menu.

`- insertItemWithTitle:atIndex:` (page 1972)
Inserts an item at the specified position in the menu.

- `removeAllItems` (page 1977)
  Removes all items in the receiver's item menu.
- `removeItemWithTitle:` (page 1979)
  Removes the item with the specified title from the menu.
- `removeItemAtIndex:` (page 1978)
  Removes the item at the specified index.

Getting the User's Selection

- `selectedItem` (page 1979)
  Returns the menu item last selected by the user.
- `titleOfSelectedItem` (page 1985)
  Returns the title of the item last selected by the user.
- `indexOfSelectedItem` (page 1971)
  Returns the index of the item last selected by the user.
- `objectValue` (page 1976)
  Returns the index of the selected item.

Setting the Current Selection

- `selectItem:` (page 1980)
  Selects the specified menu item.
- `selectItemAtIndex:` (page 1980)
  Selects the item in the menu at the specified index.
- `selectItemWithTag:` (page 1980)
  Selects the menu item with the specified tag.
- `selectItemWithTitle:` (page 1981)
  Selects the item with the specified title.
- `setObjectValue:` (page 1983)
  Selects the item at a specific index using an object value.

Getting Menu Items

- `menu` (page 1976)
  Returns the pop-up button's associated menu.
- `setMenu:` (page 1982)
  Sets the pop-up button's associated menu .
- `numberOfItems` (page 1976)
  Returns the number of items in the menu.
- `itemArray` (page 1973)
  Returns the items in the menu.
- `itemAtIndex:` (page 1973)
  Returns the menu item at the specified index.

- `itemTitleAtIndex:` (page 1974)
  Returns the title of the item at the specified index.
- `itemTitles` (page 1974)
  Returns the titles of all of the items in the menu.
- `itemWithTitle:` (page 1975)
  Returns the menu item with the specified title.
- `lastItem` (page 1975)
  Returns the last item in the menu.

Getting the Indices of Menu Items

- `indexOfItem:` (page 1969)
  Returns the index of the specified menu item.
- `indexOfItemWithTag:` (page 1969)
  Returns the index of the menu item with the specified tag.
- `indexOfItemWithTitle:` (page 1970)
  Returns the index of the item with the specified title.
- `indexOfItemWithRepresentedObject:` (page 1969)
  Returns the index of the menu item that holds the specified represented object.
- `indexOfItemWithTarget:andAction:` (page 1970)
  Returns the index of the menu item with the specified target and action.

Setting the Cell Edge to Pop out in Restricted Situations

- `preferredEdge` (page 1977)
  Returns the edge of the receiver next to which the pop-up menu is displayed under restrictive screen conditions.
- `setPreferredEdge:` (page 1983)
  Sets the edge of the receiver next to which the pop-up menu should appear under restrictive screen conditions.

Setting the Title

- `setTitle:` (page 1984)
  Sets the string displayed in the receiver when the user isn't pressing the mouse button.

Setting the Image

- `setImage:` (page 1982)
  This method has no effect.

Setting the State

- `synchronizeTitleAndSelectedItem` (page 1984)
  Ensures that the item being displayed by the receiver agrees with the selected item.

Instance Methods

addItemsWithTitles:

Adds multiple items to the end of the menu.

```
- (void)addItemsWithTitles:(NSArray *)itemTitles
```

Parameters

itemTitles
  An array of `NSString` objects containing the titles of the items you want to add. Each string in the array should be unique. If an item with the same title already exists in the menu, the existing item is removed and the new one is added.

Discussion
If you want to move an item, it's better to invoke `removeItemWithTitle:` (page 1979) explicitly and then send this method. After adding the items, this method uses the `synchronizeTitleAndSelectedItem` (page 1984) method to make sure the item being displayed matches the currently selected item.

Since this method searches for duplicate items, it should not be used if you are adding items to an already populated menu with more than a few hundred items. Add items directly to the receiver's menu instead.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `insertItemWithTitle:atIndex:` (page 1972)
- `removeAllItems` (page 1977)
- `removeItemWithTitle:` (page 1979)

Related Sample Code
AlbumToSlideshow
MovieAssembler
TimelineToTC

Declared In
`NSPopUpButton.h`

addItemWithTitle:

Adds an item with the specified title to the end of the menu.

```
- (void)addItemWithTitle:(NSString *)title
```

Parameters

title

The title of the menu-item entry. If an item with the same title already exists in the menu, the existing item is removed and the new one is added.

Discussion

If you want to move an item, it's better to invoke `removeItemWithTitle:` (page 1979) explicitly and then send this method. After adding the item, this method calls the `synchronizeTitleAndSelectedItem` (page 1984) method to make sure the item being displayed matches the currently selected item.

Since this method searches for duplicate items, it should not be used if you are adding an item to an already populated menu with more than a few hundred items. Add items directly to the receiver's menu instead.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `insertItemWithTitle:atIndex:` (page 1972)
- `removeItemWithTitle:` (page 1979)
- `setTitle:` (page 1984)

Related Sample Code

MovieAssembler
QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit
UIElementInspector
WhackedTV

Declared In

`NSPopUpButton.h`

autoenablesItems

Returns whether the receiver automatically enables and disables its items every time a user event occurs.

```
- (BOOL)autoenablesItems
```

Return Value

`YES` if the receiver automatically enables and disables items; otherwise, `NO`. The default value is `YES`.

Discussion

For more information on enabling and disabling menu items, see the NSMenuValidation protocol specification.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setAutoenablesItems:` (page 1982)

Declared In

`NSPopUpButton.h`

indexOfItem:

Returns the index of the specified menu item.

```
- (NSInteger)indexOfItem:(NSMenuItem *)anObject
```

Parameters

anObject

The menu item whose index you want.

Return Value

The index of the item or `-1` if no such item was found.

Discussion

This method invokes the method of the same name of its `NSPopUpButtonCell` object.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSPopUpButton.h`

indexOfItemWithRepresentedObject:

Returns the index of the menu item that holds the specified represented object.

```
- (NSInteger)indexOfItemWithRepresentedObject:(id)anObject
```

Parameters

anObject

The represented object associated with a menu item.

Return Value

The index of the menu item that owns the specified object, or `-1` if no such menu item was found.

Discussion

Represented objects bear some direct relation to the title or image of a menu item; for example, an item entitled "100" might have an `NSNumber` object encapsulating that value as its represented object. This method invokes the method of the same name of its `NSPopUpButtonCell` object.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

OutputBinsPDE

Declared In

`NSPopUpButton.h`

indexOfItemWithTag:

Returns the index of the menu item with the specified tag.

```
- (NSInteger)indexOfItemWithTag:(NSInteger)tag
```

Parameters

tag

The tag of the menu item you want.

Return Value

The index of the item or `-1` if no item with the specified tag was found.

Discussion

This method invokes the method of the same name of its `NSPopUpButtonCell` object.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSPopUpButton.h`

indexOfItemWithTarget:andAction:

Returns the index of the menu item with the specified target and action.

```
- (NSInteger)indexOfItemWithTarget:(id)target andAction:(SEL)actionSelector
```

Parameters

target

The target object associated with the menu item.

actionSelector

The action method associated with the menu item.

Return Value

The index of the menu item, or `-1` if no menu item contains the specified target and action.

Discussion

If you specify `NULL` for the *actionSelector* parameter, the index of the first menu item with the specified target is returned. This method invokes the method of the same name of its `NSPopUpButtonCell` object.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSPopUpButton.h`

indexOfItemWithTitle:

Returns the index of the item with the specified title.

```
- (NSInteger)indexOfItemWithTitle:(NSString *)title
```

Parameters

title

The title of the item you want.

Return Value

The index of the item or `-1` if no item with the specified title was found.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AlbumToSlideshow
QTAudioContextInsert
QTAudioExtractionPanel

Declared In
`NSPopUpButton.h`

indexOfSelectedItem

Returns the index of the item last selected by the user.

```
- (NSInteger)indexOfSelectedItem
```

Return Value
The index of the selected item, or `-1` if no item is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- selectedItem` (page 1979)
`- titleOfSelectedItem` (page 1985)

Related Sample Code
AlbumToSlideshow
CAPlayThrough
GLUT
iChatTheater
QTAudioContextInsert

Declared In
`NSPopUpButton.h`

initWithFrame:pullsDown:

Returns an `NSPopUpButton` object initialized to the specified dimensions.

```
- (id)initWithFrame:(NSRect)frameRect pullsDown:(BOOL)flag
```

Parameters

frameRect
The frame rectangle for the button, specified in the parent view's coordinate system.

flag
`YES` if you want the receiver to display a pull-down menu; otherwise, `NO` if you want it to display a pop-up menu.

Return Value
An initialized `NSPopUpButton` object, or `nil` if the object could not be initialized.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `pullsDown` (page 1977)
- `setPullsDown:` (page 1983)

Related Sample Code
ButtonMadness
MenuItemView
MenuMadness
Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility

Declared In
`NSPopUpButton.h`

insertItemWithTitle:atIndex:

Inserts an item at the specified position in the menu.

```
- (void)insertItemWithTitle:(NSString *)title atIndex:(NSInteger)index
```

Parameters

title
The title of the new item. If an item with the same title already exists in the menu, the existing item is removed and the new one is added

index
The zero-based index at which to insert the item. Specifying 0 inserts the item at the top of the menu.

Discussion
If you want to move an item, it's better to invoke `removeItemWithTitle:` (page 1979) explicitly and then send this method. After adding the item, this method uses the `synchronizeTitleAndSelectedItem` (page 1984) method to make sure the item displayed matches the currently selected item.

Since this method searches for duplicate items, it should not be used if you are adding an item to an already populated menu with more than a few hundred items. Add items directly to the receiver's menu instead.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addItemWithTitle:` (page 1967)
- `addItemsWithTitles:` (page 1967)
- `indexOfItemWithTitle:` (page 1970)
- `removeItemWithTitle:` (page 1979)

Related Sample Code
CAPlayThrough

QTAudioContextInsert
QTAudioExtractionPanel

Declared In
NSPopUpButton.h

itemArray

Returns the items in the menu.

```
- (NSArray *)itemArray
```

Return Value
An array of `id <NSMenuItem>` objects representing the items in the menu.

Discussion
Usually you access the menu's items and modify the menu using the methods of `NSPopUpButton` rather than accessing the array of items directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `itemAtIndex:` (page 1973)
- `insertItemWithTitle:atIndex:` (page 1972)
- `removeItemAtIndex:` (page 1978)

Declared In
NSPopUpButton.h

itemAtIndex:

Returns the menu item at the specified index.

```
- (NSMenuItem *)itemAtIndex:(NSInteger)index
```

Parameters
index
The index of the item you want.

Return Value
The menu item, or `nil` if no item exists at the specified index.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `itemWithTitle:` (page 1975)
- `lastItem` (page 1975)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel

Quartz Composer WWDC 2005 TextEdit
SampleScannerApp

Declared In
NSPopUpButton.h

itemTitleAtIndex:

Returns the title of the item at the specified index.

```
- (NSString *)itemTitleAtIndex:(NSInteger)index
```

Parameters

index
The index of the item you want.

Return Value
The title of the item, or an empty string if no item exists at the specified index.

Availability
Available in Mac OS X v10.0 and later.

See Also
- itemTitles (page 1974)

Declared In
NSPopUpButton.h

itemTitles

Returns the titles of all of the items in the menu.

```
- (NSArray *)itemTitles
```

Return Value
An array of NSString objects containing the titles of every item in the menu. The titles appear in the order in which the items appear in the menu.

Discussion
If the menu contains separator items, the array contains an empty string (@"") for each separator item.

Availability
Available in Mac OS X v10.0 and later.

See Also
- itemTitleAtIndex: (page 1974)
- itemWithTitle: (page 1975)
- numberOfItems (page 1976)

Declared In
NSPopUpButton.h

itemWithTitle:

Returns the menu item with the specified title.

```
- (NSMenuItem *)itemWithTitle:(NSString *)title
```

Parameters

title

The title of the menu item you want.

Return Value

The menu item, or `nil` if no item with the specified title exists in the menu.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addItemWithTitle:` (page 1967)
- `selectItemWithTitle:` (page 1981)
- `itemAtIndex:` (page 1973)
- `indexOfItemWithTitle:` (page 1970)

Declared In

`NSPopUpButton.h`

lastItem

Returns the last item in the menu.

```
- (NSMenuItem *)lastItem
```

Return Value

The last menu item.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `itemAtIndex:` (page 1973)

Related Sample Code

QTAudioContextInsert

QTAudioExtractionPanel

Quartz Composer WWDC 2005 TextEdit

UIElementInspector

WhackedTV

Declared In

`NSPopUpButton.h`

menu

Returns the pop-up button's associated menu.

```
- (NSMenu *)menu
```

Return Value
The menu for the pop-up button.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
OutputBins2PDE
QTAudioContextInsert
QTAudioExtractionPanel
Quartz Composer WWDC 2005 TextEdit
UIElementInspector

Declared In
`NSPopUpButton.h`

numberOfItems

Returns the number of items in the menu.

```
- (NSInteger)numberOfItems
```

Return Value
The number of items in the menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- lastItem` (page 1975)

Related Sample Code
MovieAssembler
QTAudioContextInsert

Declared In
`NSPopUpButton.h`

objectValue

Returns the index of the selected item.

```
- (id)objectValue
```

Return Value
An object (typically an `NSNumber` object) that responds to the `intValue` message and contains the index of the selected item.

See Also
- `setObjectValue:` (page 1983)

preferredEdge

Returns the edge of the receiver next to which the pop-up menu is displayed under restrictive screen conditions.

```
- (NSRectEdge)preferredEdge
```

Return Value
Possible values include `NSMinXEdge`, `NSMinYEdge`, `NSMaxXEdge`, or `NSMaxYEdge`. The default value is the bottom edge, which is `NSMaxYEdge` for flipped views or `NSMinYEdge` for unflipped views.

Discussion
For pull-down menus, the default behavior is to display the menu under the receiver. For most pop-up menus, the `NSPopUpButton` object attempts to show the selected item directly over the button.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPreferredEdge:` (page 1983)

Declared In
`NSPopUpButton.h`

pullsDown

Returns a Boolean value indicating the behavior of the control's menu.

```
- (BOOL)pullsDown
```

Return Value
`YES` if the menu behaves like a pull-down menu; otherwise, `NO` if it behaves like a pop-up menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPullsDown:` (page 1983)

Declared In
`NSPopUpButton.h`

removeAllItems

Removes all items in the receiver's item menu.

```
- (void)removeAllItems
```

Discussion
After removing the items, this method uses the `synchronizeTitleAndSelectedItem` (page 1984) method to refresh the menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfItems` (page 1976)
- `removeItemAtIndex:` (page 1978)
- `removeItemWithTitle:` (page 1979)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
Quartz Composer WWDC 2005 TextEdit
UIElementInspector
WhackedTV

Declared In
`NSPopUpButton.h`

removeItemAtIndex:

Removes the item at the specified index.

```
- (void)removeItemAtIndex:(NSInteger)index
```

Parameters
index
The zero-based index indicating which item to remove. Specifying 0 removes the item at the top of the menu.

Discussion
After removing the item, this method uses the `synchronizeTitleAndSelectedItem` (page 1984) method to make sure the title displayed matches the currently selected item.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `insertItemWithTitle:atIndex:` (page 1972)
- `removeAllItems` (page 1977)
- `removeItemWithTitle:` (page 1979)

Related Sample Code
CocoaSpeechSynthesisExample
QTAudioContextInsert

Declared In
`NSPopUpButton.h`

removeItemWithTitle:

Removes the item with the specified title from the menu.

```
- (void)removeItemWithTitle:(NSString *)title
```

Parameters

title

The title of the item you want to remove. If no menu item exists with the specified title, this method triggers an assertion.

Discussion

This method removes the first item it finds with the specified name. This method then uses `synchronizeTitleAndSelectedItem` (page 1984) to refresh the menu.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addItemWithTitle:` (page 1967)
- `removeAllItems` (page 1977)
- `removeItemAtIndex:` (page 1978)

Declared In

`NSPopUpButton.h`

selectedItem

Returns the menu item last selected by the user.

```
- (NSMenuItem *)selectedItem
```

Return Value

The menu item that is currently selected, or `nil` if no item is selected.

Discussion

The last selected menu item is the one that was highlighted when the user released the mouse button. It is possible for a pull-down menu's selected item to be its first item.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

FinalCutPro_AppleEvents
QTAudioContextInsert
QTAudioExtractionPanel
Quartz Composer WWDC 2005 TextEdit
UIElementInspector

Declared In

`NSPopUpButton.h`

selectItem:

Selects the specified menu item.

```
- (void)selectItem:(NSMenuItem *)item
```

Parameters

anObject

The menu item to select, or `nil` if you want to deselect all menu items.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
NewsReader
QTAudioContextInsert
QTAudioExtractionPanel
WhackedTV

Declared In
`NSPopUpButton.h`

selectItemAtIndex:

Selects the item in the menu at the specified index.

```
- (void)selectItemAtIndex:(NSInteger)index
```

Parameters

index

The index of the item you want to select, or `-1` you want to deselect all menu items.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- indexOfSelectedItem` (page 1971)

Related Sample Code
ColorMatching
GLUT
QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit
WhackedTV

Declared In
`NSPopUpButton.h`

selectItemWithTag:

Selects the menu item with the specified tag.

```
- (BOOL)selectItemWithTag:(NSInteger)tag
```

Parameters

tag

The tag of the item you want to select.

Return Value

`YES` if the item was successfully selected; otherwise, `NO`.

Discussion

If no item with the specified tag is found, this method returns `NO` and leaves the menu state unchanged.

You typically assign tags to menu items from Interface Builder, but you can also assign them programmatically using the `setTag:` method of `NSMenuItem`.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- indexOfItemWithTag:` (page 1969)

Related Sample Code

ScannerBrowser

WhackedTV

Declared In

`NSPopUpButton.h`

selectItemWithTitle:

Selects the item with the specified title.

```
- (void)selectItemWithTitle:(NSString *)title
```

Parameters

title

The title of the item to select. If you specify `nil`, an empty string, or a string that does not match the title of a menu item, this method deselects the currently selected item.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- indexOfItemWithTitle:` (page 1970)

`- addItemWithTitle:` (page 1967)

`- setTitle:` (page 1984)

Related Sample Code

GLUT

Declared In

`NSPopUpButton.h`

setAutoenablesItems:

Sets whether the receiver automatically enables and disables its items every time a user event occurs.

```
- (void)setAutoenablesItems:(BOOL)flag
```

Parameters

flag

YES if you want the receiver to automatically enable and disable items; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- autoenablesItems (page 1968)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
WhackedTV

Declared In
NSPopUpButton.h

setImage:

This method has no effect.

```
- (void)setImage:(NSImage *)anImage
```

Parameters

anImage

The image to display.

Discussion
The image displayed in a pop up button cell is taken from the selected menu item (in the case of a pop up menu) or from the first menu item (in the case of a pull-down menu).

setMenu:

Sets the pop-up button's associated menu .

```
- (void)setMenu:(NSMenu *)menu
```

Parameters

menu

The menu to associate with the pop-up button.

Discussion
If another menu was already associated with the pop-up button, this method releases the old menu. If you want to explicitly save the old menu, you should retain it before invoking this method.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ScannerBrowser

Declared In
NSPopUpButton.h

setObjectValue:

Selects the item at a specific index using an object value.

```
- (void)setObjectValue:(id)object
```

Parameters

object

An NSNumber object containing the index (an integer) of the item you want to select. Specify the index -1 to deselect all items. You can also use an object other than an NSNumber object. In that case, the object must respond to the intValue method and return an appropriate index value.

Discussion

See Also
- objectValue (page 1976)

setPreferredEdge:

Sets the edge of the receiver next to which the pop-up menu should appear under restrictive screen conditions.

```
- (void)setPreferredEdge:(NSRectEdge)edge
```

Parameters

edge

The preferred edge. Possible values include NSMinXEdge, NSMinYEdge, NSMaxXEdge, or NSMaxYEdge.

Discussion
For pull-down menus, the default behavior is to display the menu under the receiver. For most pop-up menus, the NSPopUpButton object attempts to show the selected item directly over the button.

Availability
Available in Mac OS X v10.0 and later.

See Also
- preferredEdge (page 1977)

Declared In
NSPopUpButton.h

setPullsDown:

Sets whether the receiver behaves as a pull-down or pop-up menu.

```
- (void)setPullsDown:(BOOL)flag
```

Parameters

flag

YES if you want the receiver to operate as a pull-down menu; otherwise, NO if you want it to operate as a pop-up menu.

Discussion

This method does not change the contents of the menu; it changes only the style of the menu.

When changing the menu type to a pull-down menu, if the menu was a pop-up menu and the cell alters the state of its selected items, this method sets the state of the currently selected item to NSStateOff before changing the menu type.

Availability

Available in Mac OS X v10.0 and later.

See Also

- initWithFrame:pullsDown: (page 1971)
- pullsDown (page 1977)

Declared In

NSPopUpButton.h

setTitle:

Sets the string displayed in the receiver when the user isn't pressing the mouse button.

```
- (void)setTitle:(NSString *)aString
```

Parameters

aString

The string to display.

Discussion

If the receiver displays a pop-up menu, this method changes the current item to be the item with the specified title, adding a new item by that name if one does not already exist. If the receiver displays a pull-down list, this method sets its title to the specified string.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSPopUpButton.h

synchronizeTitleAndSelectedItem

Ensures that the item being displayed by the receiver agrees with the selected item.

```
- (void)synchronizeTitleAndSelectedItem
```

Discussion

If there's no selected item, this method selects the first item in the item menu and sets the receiver's item to match. For pull-down menus, this method makes sure that the first item is being displayed (the NSPopUpButtonCell object must be set to use the selected menu item, which happens by default).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `itemArray` (page 1973)
- `indexOfSelectedItem` (page 1971)

Declared In
`NSPopUpButton.h`

titleOfSelectedItem

Returns the title of the item last selected by the user.

```
- (NSString *)titleOfSelectedItem
```

Return Value
The title of the selected menu item, or an empty string if no item is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfSelectedItem` (page 1971)

Related Sample Code
AlbumToSlideshow
FinalCutPro_AppleEvents
GLUT
MovieAssembler
WhackedTV

Declared In
`NSPopUpButton.h`

Notifications

NSPopUpButtonWillPopUpNotification

Posted when an `NSPopUpButton` object receives a mouse-down event—that is, when the user is about to select an item from the menu.

The notification object is the selected `NSPopUpButton` object. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPopUpButton.h`

NSPopUpButtonCell Class Reference

Inherits from	NSMenuItemCell : NSButtonCell : NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSPopUpButtonCell.h
Companion guide	Application Menu and Pop-up List Programming Topics
Related sample code	ButtonMadness MenuItemView MenuMadness QTAudioContextInsert Quartz Composer WWDC 2005 TextEdit

Overview

The `NSPopUpButtonCell` class defines the visual appearance of pop-up buttons that display pop-up or pull-down menus. Pop-up menus present the user with a set of choices, much the way radio buttons do, but using much less space. Pull-down menus also provide a set of choices but present the information in a slightly different way, usually to provide a set of commands from which the user can choose.

The `NSPopUpButtonCell` class implements the user interface for the `NSPopUpButton` class.

Note that while a menu is tracking, adding, removing, or changing items on the menu is not reflected.

Tasks

Initialization

`- initTextCell:pullsDown:` (page 1997)
Returns an `NSPopUpButtonCell` object initialized with the specified title.

Getting and Setting Attributes

- `setMenu:` (page 2009)
  Sets the pop-up button's associated menu.
- `menu` (page 2001)
  Returns the pop-up button's associated menu.
- `setPullsDown:` (page 2010)
  Sets whether the receiver behaves as a pull-down or pop-up menu.
- `pullsDown` (page 2003)
  Returns a Boolean value indicating the behavior of the control's menu.
- `setAutoenablesItems:` (page 2008)
  Sets whether the receiver automatically enables and disables its items every time a user event occurs.
- `autoenablesItems` (page 1993)
  Returns whether the receiver automatically enables and disables its items every time a user event occurs.
- `setPreferredEdge:` (page 2010)
  Sets the edge of the receiver next to which the pop-up menu should appear under restrictive screen conditions.
- `preferredEdge` (page 2002)
  Returns the edge of the receiver next to which the pop-up menu is displayed under restrictive screen conditions.
- `setUsesItemFromMenu:` (page 2011)
  Sets whether the pop-up button uses an item from the menu for its own title.
- `usesItemFromMenu` (page 2013)
  Returns a Boolean value indicating whether the pop-up button uses an item from the menu for its own title.
- `setAltersStateOfSelectedItem:` (page 2007)
  Sets whether the receiver links the state of the menu items to the current selection.
- `altersStateOfSelectedItem` (page 1992)
  Returns a Boolean value indicating whether the receiver links the state of the selected menu item to the current selection.
- `setArrowPosition:` (page 2008)
  Sets the position of the arrow displayed on the receiver.
- `arrowPosition` (page 1992)
  Returns the position of the arrow displayed on the receiver.

Adding and Removing Items

- `addItemWithTitle:` (page 1991)
  Adds an item with the specified title to the end of the menu.
- `addItemsWithTitles:` (page 1990)
  Adds multiple items to the end of the menu.
- `insertItemWithTitle:atIndex:` (page 1998)
  Inserts an item at the specified position in the menu.

- `removeItemWithTitle:` (page 2004)
  Removes the item with the specified title from the menu.
- `removeItemAtIndex:` (page 2004)
  Removes the item at the specified index.
- `removeAllItems` (page 2003)
  Removes all items in the receiver's item menu.

Accessing the Items

- `itemArray` (page 1999)
  Returns the items in the menu.
- `numberOfItems` (page 2001)
  Returns the number of items in the menu.
- `indexOfItem:` (page 1994)
  Returns the index of the specified menu item.
- `indexOfItemWithTitle:` (page 1996)
  Returns the index of the item with the specified title.
- `indexOfItemWithTag:` (page 1995)
  Returns the index of the menu item with the specified tag.
- `indexOfItemWithRepresentedObject:` (page 1994)
  Returns the index of the menu item that holds the specified represented object.
- `indexOfItemWithTarget:andAction:` (page 1996)
  Returns the index of the menu item with the specified target and action.
- `itemAtIndex:` (page 1999)
  Returns the menu item at the specified index.
- `itemWithTitle:` (page 2000)
  Returns the menu item with the specified title.
- `lastItem` (page 2001)
  Returns the last item in the menu.
- `objectValue` (page 2002)
  Returns the index of the selected item.
- `setObjectValue:` (page 2009) Available in Mac OS X v10.0 through Mac OS X v10.5
  Selects the item at a specific index using an object value.

Dealing with Selection

- `selectItem:` (page 2005)
  Selects the specified menu item.
- `selectItemAtIndex:` (page 2005)
  Selects the item in the menu at the specified index.
- `selectItemWithTag:` (page 2006)
  Selects the menu item with the specified tag.

- `selectItemWithTitle:` (page 2007)
  Selects the item with the specified title.
- `setTitle:` (page 2011)
  Sets the string displayed in the receiver when the user isn't pressing the mouse button.
- `selectedItem` (page 2004)
  Returns the menu item last selected by the user.
- `indexOfSelectedItem` (page 1997)
  Returns the index of the item last selected by the user.
- `synchronizeTitleAndSelectedItem` (page 2012)
  Synchronizes the the pop-up button's displayed item with the currently selected menu item.

Title Conveniences

- `itemTitleAtIndex:` (page 1999)
  Returns the title of the item at the specified index.
- `itemTitles` (page 2000)
  Returns the titles of all of the items in the menu.
- `titleOfSelectedItem` (page 2013)
  Returns the title of the item last selected by the user.

Setting the Image

- `setImage:` (page 2009)
  This method has no effect.

Handling Events and Action Messages

- `attachPopUpWithFrame:inView:` (page 1993)
  Sets up the receiver to display a menu.
- `dismissPopUp` (page 1994)
  Dismisses the pop-up button's menu by ordering its window out.
- `performClickWithFrame:inView:` (page 2002)
  Displays the receiver's menu and track mouse events in it.

Instance Methods

addItemsWithTitles:

Adds multiple items to the end of the menu.

```
- (void)addItemsWithTitles:(NSArray *)itemTitles
```

Parameters

itemTitles

An array of `NSString` objects containing the titles of the items you want to add. Each string in the array should be unique. If an item with the same title already exists in the menu, the existing item is removed and the new one is added.

Discussion

The new menu items use the pop-up button's default action and target, but you can change these using the `setAction:` and `setTarget:` methods of the corresponding `NSMenuItem` object.

If you want to move an item, it's better to invoke `removeItemWithTitle:` (page 2004) explicitly and then send this method. After adding the items, this method uses the `synchronizeTitleAndSelectedItem` (page 2012) method to make sure the item being displayed matches the currently selected item.

Since this method searches for duplicate items, it should not be used if you are adding items to an already populated menu with more than a few hundred items. Add items directly to the receiver's menu instead.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addItemWithTitle:` (page 1991)
- `setAction:` (`NSMenuItem`)
- `setTarget:` (`NSMenuItem`)

Declared In

`NSPopUpButtonCell.h`

addItemWithTitle:

Adds an item with the specified title to the end of the menu.

```
- (void)addItemWithTitle:(NSString *)title
```

Parameters

title

The title of the new menu item. If an item with the same title already exists in the menu, the existing item is removed and the new one is added.

Discussion

The menu item uses the pop-up button's default action and target, but you can change these using the `setAction:` and `setTarget:` methods of the corresponding `NSMenuItem` object.

Since this method searches for duplicate items, it should not be used if you are adding an item to an already populated menu with more than a few hundred items. Add items directly to the button's menu instead.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addItemsWithTitles:` (page 1990)
- `setAction:` (`NSMenuItem`)
- `setTarget:` (`NSMenuItem`)

Declared In
NSPopUpButtonCell.h

altersStateOfSelectedItem

Returns a Boolean value indicating whether the receiver links the state of the selected menu item to the current selection.

```
- (BOOL)altersStateOfSelectedItem
```

Return Value
YES if the selected menu item has its state set to NSOnState automatically; otherwise, NO if the state of menu items is independent of the current selection.

Discussion
This option is usually used only by pop-up menus. You typically do not alter the state of menu items in a pull-down menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectItemAtIndex: (page 2005)
- selectItemWithTitle: (page 2007)

Declared In
NSPopUpButtonCell.h

arrowPosition

Returns the position of the arrow displayed on the receiver.

```
- (NSPopUpArrowPosition)arrowPosition
```

Return Value
The arrow position.

Discussion
NSPopUpNoArrow means no arrow is displayed. NSPopUpArrowAtCenter means the arrow is vertically centered, pointing to the right, vertically centered. NSPopUpArrowAtBottom means the arrow is at the bottom, pointing downward.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setArrowPosition: (page 2008)

Declared In
NSPopUpButtonCell.h

attachPopUpWithFrame:inView:

Sets up the receiver to display a menu.

```
- (void)attachPopUpWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
```

Parameters

cellFrame

The cell's rectangle, specified in points in the coordinate system of the view in the *controlView* parameter. The menu is attached to this rectangle.

controlView

The view in which to display the pop-up button's menu.

Discussion

This call sets up the popup button cell to display a menu, which occurs in `performClickWithFrame:inView:` (page 2002). This method sets the cell's control view and then highlights and redraws the cell. It does not show the menu.

This method also posts an `NSPopUpButtonCellWillPopUpNotification` (page 2014). (The `NSPopUpButton` object sends a corresponding `NSPopUpButtonWillPopUpNotification` (page 1985).)

You normally do not call this method explicitly. It is called by the Application Kit automatically when the menu for the pop-up button is to be displayed.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- dismissPopUp` (page 1994)

Declared In

`NSPopUpButtonCell.h`

autoenablesItems

Returns whether the receiver automatically enables and disables its items every time a user event occurs.

```
- (BOOL)autoenablesItems
```

Return Value

`YES` if the receiver automatically enables and disables items; otherwise, `NO`. The default value is `YES`.

Discussion

For more information on enabling and disabling menu items, see the NSMenuValidation (page 3729) .

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setAutoenablesItems:` (page 2008)

Declared In

`NSPopUpButtonCell.h`

dismissPopUp

Dismisses the pop-up button's menu by ordering its window out.

```
- (void)dismissPopUp
```

Discussion
If the pop-up button was not displaying its menu, this method does nothing.

You normally do not call this method explicitly. It is called by the Application Kit automatically to dismiss the menu for the pop-up button.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `attachPopUpWithFrame:inView:` (page 1993)
- `orderOut:` (page 3352) (`NSWindow`)

Declared In
`NSPopUpButtonCell.h`

indexOfItem:

Returns the index of the specified menu item.

```
- (NSInteger)indexOfItem:(NSMenuItem *)item
```

Parameters
item
The menu item whose index you want.

Return Value
The index of the item or `-1` if no such item was found.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfItemWithRepresentedObject:` (page 1994)
- `indexOfItemWithTag:` (page 1995)
- `indexOfItemWithTarget:andAction:` (page 1996)
- `indexOfItemWithTitle:` (page 1996)
- `indexOfSelectedItem` (page 1997)

Declared In
`NSPopUpButtonCell.h`

indexOfItemWithRepresentedObject:

Returns the index of the menu item that holds the specified represented object.

```
- (NSInteger)indexOfItemWithRepresentedObject:(id)obj
```

Parameters

obj

The represented object associated with a menu item.

Return Value

The index of the menu item that owns the specified object, or `-1` if no such menu item was found.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `indexOfItem:` (page 1994)
- `indexOfItemWithTag:` (page 1995)
- `indexOfItemWithTarget:andAction:` (page 1996)
- `indexOfItemWithTitle:` (page 1996)
- `indexOfSelectedItem` (page 1997)
- `representedObject` (NSMenuItem)
- `setRepresentedObject:` (NSMenuItem)

Declared In

`NSPopUpButtonCell.h`

indexOfItemWithTag:

Returns the index of the menu item with the specified tag.

```
- (NSInteger)indexOfItemWithTag:(NSInteger)tag
```

Parameters

tag

The tag of the menu item you want.

Return Value

The index of the item or `-1` if no item with the specified tag was found.

Discussion

Tags are values your application assigns to an object to identify it. You can assign tags to menu items using the `setTag:` method of `NSMenuItem`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `indexOfItem:` (page 1994)
- `indexOfItemWithRepresentedObject:` (page 1994)
- `indexOfItemWithTarget:andAction:` (page 1996)
- `indexOfItemWithTitle:` (page 1996)
- `indexOfSelectedItem` (page 1997)
- `setTag:` (`NSMenuItem`)

Declared In

`NSPopUpButtonCell.h`

indexOfItemWithTarget:andAction:

Returns the index of the menu item with the specified target and action.

```
- (NSInteger)indexOfItemWithTarget:(id)target andAction:(SEL)actionSelector
```

Parameters

target

The target object associated with the menu item.

actionSelector

The action method associated with the menu item.

Return Value

The index of the menu item, or `-1` if no menu item contains the specified target and action.

Discussion

If you specify `NULL` for the *actionSelector* parameter, the index of the first menu item with the specified target is returned.

The `NSPopUpButtonCell` class assigns a default action and target to each menu item, but you can change these values using the `setAction:` and `setTarget:` methods of `NSMenuItem`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `indexOfItem:` (page 1994)
- `indexOfItemWithRepresentedObject:` (page 1994)
- `indexOfItemWithTag:` (page 1995)
- `indexOfItemWithTarget:andAction:` (page 1996)
- `indexOfItemWithTitle:` (page 1996)
- `indexOfSelectedItem` (page 1997)
- `setAction:` (`NSMenuItem`)
- `setTarget:` (`NSMenuItem`)

Declared In

`NSPopUpButtonCell.h`

indexOfItemWithTitle:

Returns the index of the item with the specified title.

```
- (NSInteger)indexOfItemWithTitle:(NSString *)title
```

Parameters

title

The title of the item you want. You must not pass `nil` for this parameter.

Return Value

The index of the item or `-1` if no item with the specified title was found.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `indexOfItem:` (page 1994)
- `indexOfItemWithRepresentedObject:` (page 1994)
- `indexOfItemWithTag:` (page 1995)
- `indexOfItemWithTarget:andAction:` (page 1996)
- `indexOfItemWithTitle:` (page 1996)
- `indexOfSelectedItem` (page 1997)

Declared In
`NSPopUpButtonCell.h`

indexOfSelectedItem

Returns the index of the item last selected by the user.

```
- (NSInteger)indexOfSelectedItem
```

Return Value
The index of the selected item, or `-1` if no item is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfItem:` (page 1994)
- `indexOfItemWithRepresentedObject:` (page 1994)
- `indexOfItemWithTag:` (page 1995)
- `indexOfItemWithTarget:andAction:` (page 1996)
- `indexOfItemWithTitle:` (page 1996)

Declared In
`NSPopUpButtonCell.h`

initTextCell:pullsDown:

Returns an `NSPopUpButtonCell` object initialized with the specified title.

```
- (id)initTextCell:(NSString *)stringValue pullsDown:(BOOL)pullDown
```

Parameters

stringValue
The title of the first menu. You may specify an empty string if you do not want to add an initial menu item.

pullDown
`YES` if you want the receiver to display a pull-down menu; otherwise, `NO` if you want it to display a pop-up menu.

Return Value
An initialized `NSPopUpButtonCell` object, or `nil` if the object could not be initialized.

Discussion
This menu item is assigned the default pop-up button action that displays the menu. To set the action and target, use the `setAction:` and `setTarget:` methods of the item's corresponding `NSMenuItem` object.

This method is the designated initializer of the class.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAction:` (`NSMenuItem`)
`- setTarget:` (`NSMenuItem`)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel

Declared In
`NSPopUpButtonCell.h`

insertItemWithTitle:atIndex:

Inserts an item at the specified position in the menu.

```
- (void)insertItemWithTitle:(NSString *)title atIndex:(NSInteger)index
```

Parameters

title
The title of the new item. If an item with the same title already exists in the menu, the existing item is removed and the new one is added

index
The zero-based index at which to insert the item. Specifying 0 inserts the item at the top of the menu.

Discussion
The value in *index* must represent a valid position in the array. The menu item at *index* and all those that follow it are shifted down one slot to make room for the new menu item.

This method assigns the pop-up button's default action and target to the new menu item. Use the menu item's `setAction:` and `setTarget:` methods to assign a new action and target.

Since this method searches for duplicate items, it should not be used if you are adding an item to an already populated menu with more than a few hundred items. Add items directly to the button's menu instead.

Availability
Available in Mac OS X v10.0 and later.

See Also
`insertObject:atIndex:` (`NSMutableArray`)
`- setAction:` (`NSMenuItem`)
`- setTarget:` (`NSMenuItem`)

Declared In
`NSPopUpButtonCell.h`

itemArray

Returns the items in the menu.

```
- (NSArray *)itemArray
```

Return Value
An array of `NSMenuItem` objects representing the items in the menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `itemArray` (page 1620) (NSMenu)

Declared In
`NSPopUpButtonCell.h`

itemAtIndex:

Returns the menu item at the specified index.

```
- (NSMenuItem *)itemAtIndex:(NSInteger)index
```

Parameters
index
The index of the item you want. The specified index must refer to an existing menu item.

Return Value
The menu item, or `nil` if no item exists at the specified index.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `itemTitleAtIndex:` (page 1999)
- `itemAtIndex:` (page 1620) (`NSMenu`)

Declared In
`NSPopUpButtonCell.h`

itemTitleAtIndex:

Returns the title of the item at the specified index.

```
- (NSString *)itemTitleAtIndex:(NSInteger)index
```

Parameters
index
The index of the item you want.

Return Value
The title of the item, or an empty string if no item exists at the specified index.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `itemAtIndex:` (page 1999)

Declared In
`NSPopUpButtonCell.h`

itemTitles

Returns the titles of all of the items in the menu.

```
- (NSArray *)itemTitles
```

Return Value
An array of `NSString` objects containing the titles of every item in the menu. The titles appear in the order in which the items appear in the menu.

Discussion
If the menu contains separator items, the array contains an empty string (@"") for each separator item.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `itemTitleAtIndex:` (page 1999)

Declared In
`NSPopUpButtonCell.h`

itemWithTitle:

Returns the menu item with the specified title.

```
- (NSMenuItem *)itemWithTitle:(NSString *)title
```

Parameters
title
The title of the menu item you want.

Return Value
The menu item, or `nil` if no item with the specified title exists in the menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `itemTitleAtIndex:` (page 1999)
- `itemAtIndex:` (page 1999)

Declared In
`NSPopUpButtonCell.h`

lastItem

Returns the last item in the menu.

```
- (NSMenuItem *)lastItem
```

Return Value
The last menu item.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPopUpButtonCell.h

menu

Returns the pop-up button's associated menu.

```
- (NSMenu *)menu
```

Return Value
The menu for the pop-up button.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setMenu: (page 2009)

Declared In
NSPopUpButtonCell.h

numberOfItems

Returns the number of items in the menu.

```
- (NSInteger)numberOfItems
```

Return Value
The number of items in the menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
count (NSArray)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSPopUpButtonCell.h

objectValue

Returns the index of the selected item.

```
- (id)objectValue
```

Return Value
An object (typically an `NSNumber` object) that responds to the `intValue` message and contains the index of the selected item.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setObjectValue:` (page 2009)

Declared In
`NSPopUpButtonCell.h`

performClickWithFrame:inView:

Displays the receiver's menu and track mouse events in it.

```
- (void)performClickWithFrame:(NSRect)frame inView:(NSView *)controlView
```

Parameters

frame
The cell's rectangle, specified in points in the coordinate system of the view in the *controlView* parameter.

controlView
The view in which to display the pop-up button's menu.

Discussion
You normally do not call this method explicitly. It is called by the Application Kit automatically to handle events in the pop-up button.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- attachPopUpWithFrame:inView:` (page 1993)

Declared In
`NSPopUpButtonCell.h`

preferredEdge

Returns the edge of the receiver next to which the pop-up menu is displayed under restrictive screen conditions.

```
- (NSRectEdge)preferredEdge
```

Return Value
Possible values include `NSMinXEdge`, `NSMinYEdge`, `NSMaxXEdge`, or `NSMaxYEdge`. If no preferred edge was explicitly set, the default value is the bottom edge, which is `NSMaxYEdge` for flipped views or `NSMinYEdge` for unflipped views.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPreferredEdge:` (page 2010)

Declared In
`NSPopUpButtonCell.h`

pullsDown

Returns a Boolean value indicating the behavior of the control's menu.

```
- (BOOL)pullsDown
```

Return Value
`YES` if the menu behaves like a pull-down menu; otherwise, `NO` if it behaves like a pop-up menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPullsDown:` (page 2010)

Declared In
`NSPopUpButtonCell.h`

removeAllItems

Removes all items in the receiver's item menu.

```
- (void)removeAllItems
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `removeItemAtIndex:` (page 2004)
- `removeItemWithTitle:` (page 2004)
- `insertItemWithTitle:atIndex:` (page 1998)

Declared In
`NSPopUpButtonCell.h`

removeItemAtIndex:

Removes the item at the specified index.

```
- (void)removeItemAtIndex:(NSInteger)index
```

Parameters

index

The zero-based index indicating which item to remove. Specifying 0 removes the item at the top of the menu. The index must be valid and non-negative.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `removeAllItems` (page 2003)
- `removeItemWithTitle:` (page 2004)
- `insertItemWithTitle:atIndex:` (page 1998)

Declared In
`NSPopUpButtonCell.h`

removeItemWithTitle:

Removes the item with the specified title from the menu.

```
- (void)removeItemWithTitle:(NSString *)title
```

Parameters

title

The title of the item you want to remove. If no menu item exists with the specified title, this method triggers an assertion.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `removeAllItems` (page 2003)
- `removeItemAtIndex:` (page 2004)
- `insertItemWithTitle:atIndex:` (page 1998)

Declared In
`NSPopUpButtonCell.h`

selectedItem

Returns the menu item last selected by the user.

```
- (NSMenuItem *)selectedItem
```

Return Value
The menu item that is currently selected, or `nil` if no item is selected.

Discussion
The last selected menu item is the one that was highlighted when the user released the mouse button. It is possible for a pull-down menu's selected item to be its first item.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectItem:` (page 2005)
- `selectItemAtIndex:` (page 2005)
- `selectItemWithTitle:` (page 2007)

Declared In
`NSPopUpButtonCell.h`

selectItem:

Selects the specified menu item.

```
- (void)selectItem:(NSMenuItem *)item
```

Parameters

item

The menu item to select, or `nil` if you want to deselect all menu items.

Discussion
By default, selecting or deselecting a menu item from a pop-up menu changes its state. Selecting a menu item from a pull-down menu does not automatically alter the state of the item. Use the `setAltersStateOfSelectedItem:` (page 2007) method, passing it a value of `NO`, to disassociate the current selection from the state of menu items.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedItem` (page 2004)
- `selectItemAtIndex:` (page 2005)
- `selectItemWithTitle:` (page 2007)
- `setAltersStateOfSelectedItem:` (page 2007)
- `setState:` (`NSMenuItem`)

Declared In
`NSPopUpButtonCell.h`

selectItemAtIndex:

Selects the item in the menu at the specified index.

```
- (void)selectItemAtIndex:(NSInteger)index
```

Parameters

index

The index of the item you want to select, or `-1` you want to deselect all menu items.

Discussion

By default, selecting or deselecting a menu item from a pop-up menu changes its state. Selecting a menu item from a pull-down menu does not automatically alter the state of the item. Use the `setAltersStateOfSelectedItem:` (page 2007) method, passing it a value of `NO`, to disassociate the current selection from the state of menu items.

Subclassers can override this method to catch all select calls.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `selectedItem` (page 2004)
- `selectItem:` (page 2005)
- `selectItemWithTitle:` (page 2007)
- `setAltersStateOfSelectedItem:` (page 2007)
- `setState:` (`NSMenuItem`)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

`NSPopUpButtonCell.h`

selectItemWithTag:

Selects the menu item with the specified tag.

```
- (BOOL)selectItemWithTag:(NSInteger)tag
```

Parameters

tag

The tag of the item you want to select.

Return Value

`YES` if the item was successfully selected; otherwise, `NO`.

Discussion

If no item with the specified tag is found, this method returns `NO` and leaves the menu state unchanged.

You typically assign tags to menu items from Interface Builder, but you can also assign them programmatically using the `setTag:` method of `NSMenuItem`.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSPopUpButtonCell.h`

selectItemWithTitle:

Selects the item with the specified title.

```
- (void)selectItemWithTitle:(NSString *)title
```

Parameters

title

The title of the item to select. If you specify `nil`, an empty string, or a string that does not match the title of a menu item, this method deselects the currently selected item.

Discussion

By default, selecting or deselecting a menu item changes its state. Use the `setAltersStateOfSelectedItem:` (page 2007) method, passing it a value of `NO`, to disassociate the current selection from the state of menu items.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `selectedItem` (page 2004)
- `selectItem:` (page 2005)
- `selectItemAtIndex:` (page 2005)
- `setAltersStateOfSelectedItem:` (page 2007)
- `setState:` (`NSMenuItem`)

Declared In

`NSPopUpButtonCell.h`

setAltersStateOfSelectedItem:

Sets whether the receiver links the state of the menu items to the current selection.

```
- (void)setAltersStateOfSelectedItem:(BOOL)flag
```

Parameters

flag

`YES` if the selected menu item has its state set to `NSOnState` automatically; otherwise, `NO` if the state of menu items is independent of the current selection.

Discussion

You use this method to control whether the selected menu item is linked to the state of that item. When you specify `NO` for the *flag* parameter, this method sets the state of the currently selected item to `NSOffState`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `altersStateOfSelectedItem` (page 1992)
- `selectedItem` (page 2004)
- `selectItem:` (page 2005)
- `selectItemAtIndex:` (page 2005)
- `setState:` (`NSMenuItem`)

Declared In
NSPopUpButtonCell.h

setArrowPosition:

Sets the position of the arrow displayed on the receiver.

```
- (void)setArrowPosition:(NSPopUpArrowPosition)position
```

Parameters

position
The position of the arrow.

Discussion
If you specify NSPopUpNoArrow, the receiver displays no arrow. NSPopUpArrowAtCenter displays the arrow centered horizontally within the cell. NSPopUpArrowAtBottom displays the arrow at the edge of the cell. This method works with setPreferredEdge: to determine the exact location and orientation of the arrow. For more information, see setPreferredEdge: (page 2010).

This method is ignored unless the receiver is a pull-down list with a beveled border.

Availability
Available in Mac OS X v10.0 and later.

See Also
- arrowPosition (page 1992)

Related Sample Code
Sketch+Accessibility

Declared In
NSPopUpButtonCell.h

setAutoenablesItems:

Sets whether the receiver automatically enables and disables its items every time a user event occurs.

```
- (void)setAutoenablesItems:(BOOL)flag
```

Parameters

flag
YES if you want the receiver to automatically enable and disable items; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- autoenablesItems (page 1993)

Declared In
NSPopUpButtonCell.h

setImage:

This method has no effect.

```
- (void)setImage:(NSImage *)anImage
```

Parameters

anImage

The image to display.

Discussion

The image displayed in a pop up button is taken from the selected menu item (in the case of a pop up menu) or from the first menu item (in the case of a pull-down menu).

setMenu:

Sets the pop-up button's associated menu.

```
- (void)setMenu:(NSMenu *)menu
```

Parameters

menu

The menu to associate with the pop-up button.

Discussion

If another menu was already associated with the pop-up button, this method releases the old menu. If you want to explicitly save the old menu, you should retain it before invoking this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `menu` (page 2001)

Related Sample Code

QTAudioContextInsert

QTAudioExtractionPanel

Declared In

`NSPopUpButtonCell.h`

setObjectValue:

Selects the item at a specific index using an object value.

```
- (void)setObjectValue:(id)object
```

Parameters

object

An `NSNumber` object containing the index (an integer) of the item you want to select. Specify the index -1 to deselect all items. You can also use an object other than an `NSNumber` object. In that case, the object must respond to the `intValue` message and return an appropriate index value.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `objectValue` (page 2002)

Declared In
`NSPopUpButtonCell.h`

setPreferredEdge:

Sets the edge of the receiver next to which the pop-up menu should appear under restrictive screen conditions.

```
- (void)setPreferredEdge:(NSRectEdge)edge
```

Parameters

edge

The preferred edge. Possible values include `NSMinXEdge`, `NSMinYEdge`, `NSMaxXEdge`, or `NSMaxYEdge`.

Discussion
At display time, if attaching the menu to the preferred edge would cause part of the menu to be obscured, the pop-up button may use a different edge. If no preferred edge is set, the pop-up button uses the bottom edge by default.

This method works with `setArrowPosition:` (page 2008) to determine the exact location of the arrow:

- If the arrow position is `NSPopUpArrowAtCenter`, the arrow stays in the center of the button and this method determines which edge the arrow points to. `NSMinXEdge` points to the left, `NSMaxYEdge` points to the top, `NSMaxXEdge` points to the right, and `NSMinYEdge` points to the bottom.
- If the arrow position is `NSPopUpArrowAtBottom`, this method determines which edge the arrow is at. `NSMinXEdge` places the arrow at the center of the left side, pointing to the left. `NSMinYEdge` places the arrow at bottom right corner, pointing up. `NSMaxXEdge` places the arrow at the center of the right side, pointing to the right. `NSMaxYEdge` places the arrow at the bottom right corner, pointing down.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `preferredEdge` (page 2002)

Related Sample Code
ButtonMadness

Declared In
`NSPopUpButtonCell.h`

setPullsDown:

Sets whether the receiver behaves as a pull-down or pop-up menu.

```
- (void)setPullsDown:(BOOL)flag
```

Parameters

flag

YES if you want the receiver to operate as a pull-down menu; otherwise, NO if you want it to operate as a pop-up menu.

Discussion

This method does not change the contents of the menu; it changes only the style of the menu.

When changing the menu type to a pull-down menu, if the menu was a pop-up menu and the cell alters the state of its selected items, this method sets the state of the currently selected item to NSOffState before changing the menu type.

Availability

Available in Mac OS X v10.0 and later.

See Also

- pullsDown (page 2003)
- synchronizeTitleAndSelectedItem (page 2012)

Related Sample Code

ButtonMadness

Declared In

NSPopUpButtonCell.h

setTitle:

Sets the string displayed in the receiver when the user isn't pressing the mouse button.

```
- (void)setTitle:(NSString *)aString
```

Parameters

aString

The string to display.

Discussion

For pull-down menus that get their titles from a menu item, this method simply sets the pop-up button cell's menu item to the first item in the menu. For pop-up menus, if a menu item whose title matches *aString* exists, this method makes that menu item the current selection; otherwise, it creates a new menu item with the title *aString*, adds it to the pop-up menu, and selects it.

Availability

Available in Mac OS X v10.0 and later.

See Also

- initTextCell:pullsDown: (page 1997)

Declared In

NSPopUpButtonCell.h

setUsesItemFromMenu:

Sets whether the pop-up button uses an item from the menu for its own title.

```
- (void)setUsesItemFromMenu:(BOOL)flag
```

Parameters

flag

`YES` if the button should use the first menu item as its own title; otherwise, `NO`. `YES` is the default value.

Discussion

For pop-up menus, the pop-up button uses the title of the currently selected menu item; if no menu item is selected, the pop-up button displays no item and is drawn empty. You can set the title or image of the pop-up button to something permanent by first calling this method (with a parameter of `NO`) and then calling `setMenuItem:` (page 1681), as shown in the following example:

```
- (void)awakeFromNib {

    NSString *buttonImagePath = [[NSBundle mainBundle] pathForResource:@"plane"
 ofType:@"png"];
    NSImage *buttonImage = [[NSImage alloc]
initWithContentsOfFile:buttonImagePath];
    NSMenuItem *imageItem = [[NSMenuItem alloc] init];
    [imageItem setImage:buttonImage];
    [imageItem setTitle:@"City"];
    [[myPopUpButton cell] setUsesItemFromMenu:NO];
    [[myPopUpButton cell] setMenuItem:imageItem];
    [buttonImage release];
    [imageItem release];
}
```

Availability

Available in Mac OS X v10.0 and later.

See Also

`- usesItemFromMenu` (page 2013)

Declared In

`NSPopUpButtonCell.h`

synchronizeTitleAndSelectedItem

Synchronizes the the pop-up button's displayed item with the currently selected menu item.

```
- (void)synchronizeTitleAndSelectedItem
```

Discussion

If no item is currently selected, this method synchronizes the pop-up buttons displayed item with the first menu item. If the pop-up button cell does not get its displayed item from a menu item, this method does nothing.

For pull-down menus, this method sets the displayed item to the title first menu item.

If the pop-up button's menu does not contain any menu items, this method sets the pop-up button's displayed item to `nil`, resulting in nothing being displayed in the control.

Availability

Available in Mac OS X v10.0 and later.

Declared In
NSPopUpButtonCell.h

titleOfSelectedItem

Returns the title of the item last selected by the user.

```
- (NSString *)titleOfSelectedItem
```

Return Value
The title of the selected menu item, or an empty string if no item is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectItemWithTitle: (page 2007)

Declared In
NSPopUpButtonCell.h

usesItemFromMenu

Returns a Boolean value indicating whether the pop-up button uses an item from the menu for its own title.

```
- (BOOL)usesItemFromMenu
```

Return Value
YES if the button uses the first menu item as its own title; otherwise, NO. YES is the default value.

Discussion
If this option is set, pull-down menus use the title of the first menu item, while pop-up menus use the title of the currently selected menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setUsesItemFromMenu: (page 2011)

Declared In
NSPopUpButtonCell.h

Constants

NSPopUpArrowPosition

These constants are defined for use with the arrowPosition (page 1992) and setArrowPosition: (page 2008) methods.

```
typedef enum {
    NSPopUpNoArrow       = 0,
    NSPopUpArrowAtCenter = 1,
    NSPopUpArrowAtBottom = 2
} NSPopUpArrowPosition;
```

Constants

`NSPopUpNoArrow`

Does not display any arrow in the receiver.

Available in Mac OS X v10.0 and later.

Declared in `NSPopUpButtonCell.h`.

`NSPopUpArrowAtCenter`

Arrow is centered vertically, pointing toward the `preferredEdge` (page 2002).

Available in Mac OS X v10.0 and later.

Declared in `NSPopUpButtonCell.h`.

`NSPopUpArrowAtBottom`

Arrow is drawn at the edge of the button, pointing toward the `preferredEdge` (page 2002).

Available in Mac OS X v10.0 and later.

Declared in `NSPopUpButtonCell.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPopUpButtonCell.h`

Notifications

NSPopUpButtonCellWillPopUpNotification

This notification is posted just before an pop-up menu is attached to its window frame. You can use this notification to lazily construct your part's menus, thus preventing unnecessary calculations until they are needed. The notification object can be either a pop-up button or its enclosed pop-up button cell. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPopUpButtonCell.h`

NSPredicateEditor Class Reference

Inherits from	NSRuleEditor : NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSPredicateEditor.h
Companion guides	Control and Cell Programming Topics for Cocoa Predicate Programming Guide
Related sample code	PhotoSearch PredicateEditorSample

Overview

`NSPredicateEditor` is a subclass of `NSRuleEditor` that is specialized for editing `NSPredicate` objects.

`NSPredicateEditor` provides an `NSPredicate` property—`objectValue` (page 824) (inherited from `NSControl`)—that you can get and set directly, and that you can bind using Cocoa bindings (you typically configure a predicate editor in Interface Builder). `NSPredicateEditor` depends on another class, `NSPredicateEditorRowTemplate`, that describes the available predicates and how to display them.

Unlike `NSRuleEditor`, `NSPredicateEditor` does not depend on its delegate to populate its rows (and *does not call the populating delegate methods*). Instead, its rows are populated from its `objectValue` property (an instance of `NSPredicate`). `NSPredicateEditor` relies on instances `NSPredicateEditorRowTemplate`, which are responsible for mapping back and forth between the displayed view values and various predicates.

`NSPredicateEditor` exposes one property, `rowTemplates` (page 2016), which is an array of `NSPredicateEditorRowTemplate` objects.

Tasks

Managing Row Templates

- `setRowTemplates:` (page 2016)
  Sets the row templates for the receiver.
- `rowTemplates` (page 2016)
  Returns the row templates for the receiver.

Instance Methods

rowTemplates

Returns the row templates for the receiver.

```
- (NSArray *)rowTemplates
```

Return Value
The row templates for the receiver.

Discussion
Until otherwise set, this contains a single compound `NSPredicateEditorRowTemplate` object.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setRowTemplates:` (page 2016)

Declared In
`NSPredicateEditor.h`

setRowTemplates:

Sets the row templates for the receiver.

```
- (void)setRowTemplates:(NSArray *)rowTemplates
```

Parameters
`rowTemplates`
An array of `NSPredicateEditorRowTemplate` objects.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `rowTemplates` (page 2016)

Declared In
`NSPredicateEditor.h`

NSPredicateEditorRowTemplate Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSPredicateEditorRowTemplate.h
Companion guides	Control and Cell Programming Topics for Cocoa Predicate Programming Guide
Related sample code	PhotoSearch PredicateEditorSample

Overview

`NSPredicateEditorRowTemplate` describes available predicates and how to display them.

You can create instances of `NSPredicateEditorRowTemplate` programmatically or in Interface Builder. By default, a non-compound row template has three views: a popup (or static text field) on the left, a popup or static text field for operators, and either a popup or other view on the right. You can subclass `NSPredicateEditorRowTemplate` to create a row template with different numbers or types of views.

`NSPredicateEditorRowTemplate` is a concrete class, but it has five primitive methods which are called by `NSPredicateEditor`: `templateViews` (page 2027), `matchForPredicate:` (page 2025), `setPredicate:` (page 2027), `displayableSubpredicatesOfPredicate:` (page 2022), and `predicateWithSubpredicates:` (page 2026). `NSPredicateEditorRowTemplate` implements all of them, but you can override them for custom templates. The primitive methods are used by an instance of `NSPredicateEditor` as follows.

First, an instance of `NSPredicateEditor` is created, and some row templates are set on it—either through a nib file or programmatically. The first thing predicate editor does is ask each of the templates for their views, using `templateViews` (page 2027).

After setting up the predicate editor, you typically send it a `setObjectValue:` (page 836) message to restore a saved predicate. `NSPredicateEditor` needs to determine which of its templates should display each predicate in the predicate tree. It does this by sending each of its row templates a `matchForPredicate:` (page 2025) message and choosing the one that returns the highest value.

After finding the best match for a predicate, `NSPredicateEditor` copies that template to get fresh views, inserts them into the proper row, and then sets the predicate on the template using `setPredicate:` (page 2027). Within that method, the `NSPredicateEditorRowTemplate` object must set its views' values to represent that predicate.

`NSPredicateEditorRowTemplate` next asks the template for the "displayable sub-predicates" of the predicate by sending a `displayableSubpredicatesOfPredicate:` (page 2022) message. If a template represents a predicate in its entirety, or if the predicate has no subpredicates, it can return `nil` for this. Otherwise, it should return a list of predicates to be made into sub-rows of that template's row. The whole process repeats for each sub-predicate.

At this point, the user sees the predicate that was saved. If the user then makes some changes to the views of the templates, this causes `NSPredicateEditor` to recompute its predicate by asking each of the templates to return the predicate represented by the new view values, passing in the subpredicates represented by the sub-rows (an empty array if there are none, or `nil` if they aren't supported by that predicate type):

`predicateWithSubpredicates:` (page 2026)

Tasks

Initializing a Template

`- initWithLeftExpressions:rightExpressions:modifier:operators:options:` (page 2024)
Initializes and returns a "pop-up-pop-up-pop-up"-style row template.

`- initWithLeftExpressions:rightExpressionAttributeType:modifier:operators:options:` (page 2023)
Initializes and returns a "pop-up-pop-up-view"-style row template.

`- initWithCompoundTypes:` (page 2022)
Initializes and returns a row template suitable for displaying compound predicates.

Core Data Integration

`+ templatesWithAttributeKeyPaths:inEntityDescription:` (page 2021)
Returns an array of predicate templates for the given attribute key paths for a given entity.

Primitive Methods

`- matchForPredicate:` (page 2025)
Returns a positive number if the receiver can represent a given predicate, and `0` if it cannot.

`- templateViews` (page 2027)
Returns the views for the receiver.

`- setPredicate:` (page 2027)
Sets the value of the views according to the given predicate.

`- displayableSubpredicatesOfPredicate:` (page 2022)
Returns the subpredicates that should be made sub-rows of a given predicate.

- `predicateWithSubpredicates:` (page 2026)
  Returns the predicate represented by the receiver's views' values and the given sub-predicates.

Information About a Row Template

- `leftExpressions` (page 2024)
  Returns the left hand expressions for the receiver.
- `rightExpressions` (page 2027)
  Returns the right hand expressions for the receiver.
- `compoundTypes` (page 2022)
  Returns the compound predicate types for the receiver.
- `modifier` (page 2025)
  Returns the comparison predicate modifier for the receiver.
- `operators` (page 2025)
  Returns the array of operators for the receiver.
- `options` (page 2026)
  Returns the comparison predicate options for the receiver.
- `rightExpressionAttributeType` (page 2026)
  Returns the attribute type of the receiver's right expression.

Class Methods

templatesWithAttributeKeyPaths:inEntityDescription:

Returns an array of predicate templates for the given attribute key paths for a given entity.

```
+ (NSArray *)templatesWithAttributeKeyPaths:(NSArray *)keyPaths
    inEntityDescription:(NSEntityDescription *)entityDescription
```

Parameters

keyPaths
  An array of attribute key paths originating at *entityDescription*. The key paths may cross relationships but must terminate in attributes.

entityDescription
  A Core Data entity description.

Return Value
An array of predicate templates for *keyPaths* originating at *entityDescription*.

Discussion
This method determines which key paths in the entity description can use the same views (that is, share the same attribute type). For each of these groups, it instantiates individual templates via `initWithLeftExpressions:rightExpressions:modifier:operators:options:` (page 2024).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

Instance Methods

compoundTypes

Returns the compound predicate types for the receiver.

```
- (NSArray *)compoundTypes
```

Return Value
An array of `NSNumber` objects specifying compound predicate types. See `Compound_Predicate_Types` for possible values.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

displayableSubpredicatesOfPredicate:

Returns the subpredicates that should be made sub-rows of a given predicate.

```
- (NSArray *)displayableSubpredicatesOfPredicate:(NSPredicate *)predicate
```

Parameters
predicate
A predicate object.

Return Value
The subpredicates that should be made sub-rows of *predicate*. For compound predicates (instances of `NSCompoundPredicate`), the array of subpredicates; for other types of predicate, returns `nil`. If a template represents a predicate in its entirety, or if the predicate has no subpredicates, returns `nil`.

Discussion
You can override this method to create custom templates that handle complicated compound predicates.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

initWithCompoundTypes:

Initializes and returns a row template suitable for displaying compound predicates.

```
- (id)initWithCompoundTypes:(NSArray *)compoundTypes
```

Parameters

compoundTypes

An array of `NSNumber` objects specifying compound predicate types. See `Compound_Predicate_Types` for possible values.

Return Value

A row template initialized for displaying compound predicates of the types specified by *compoundTypes*.

Discussion

`NSPredicateEditor` contains such a template by default.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSPredicateEditorRowTemplate.h`

initWithLeftExpressions:rightExpressionAttributeType:modifier:operators:options:

Initializes and returns a "pop-up-pop-up-view"-style row template.

```
- (id)initWithLeftExpressions:(NSArray *)leftExpressions
    rightExpressionAttributeType:(NSAttributeType)attributeType
    modifier:(NSComparisonPredicateModifier)modifier operators:(NSArray *)operators
     options:(NSUInteger)options
```

Parameters

leftExpressions

An array of `NSExpression` objects that represent the left hand side of a predicate.

attributeType

An attribute type for the right hand side of a predicate. This value dictates the type of view created, and how the control's object value is coerced before putting it into a predicate.

modifier

A modifier for the predicate (see `NSComparisonPredicateModifier` for possible values).

operators

An array of `NSNumber` objects specifying the operator type (see `NSPredicateOperatorType` for possible values).

options

Options for the predicate (see `NSComparisonPredicate_Options` for possible values).

Return Value

A row template initialized using the given arguments.

Discussion

The type of *attributeType* dictates the type of view created. For example, `NSDateAttributeType` will create an `NSDatePicker` object, `NSInteger64AttributeType` will create a short text field, and `NSStringAttributeType` will produce a longer text field. You can resize the views as you want.

Predicates do not automatically coerce types for you. For example, comparing a number to a string will raise an exception. Therefore, the attribute type is also needed to determine how the control's object value must be coerced before putting it into a predicate.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

initWithLeftExpressions:rightExpressions:modifier:operators:options:

Initializes and returns a “pop-up-pop-up-pop-up”-style row template.

```
- (id)initWithLeftExpressions:(NSArray *)leftExpressions rightExpressions:(NSArray
    *)rightExpressions modifier:(NSComparisonPredicateModifier)modifier
   operators:(NSArray *)operators options:(NSUInteger)options
```

Parameters

leftExpressions
An array of `NSExpression` objects that represent the left hand side of a predicate.

rightExpressions
An array of `NSExpression` objects that represent the right hand side of a predicate.

modifier
A modifier for the predicate (see `NSComparisonPredicateModifier` for possible values).

operators
An array of `NSNumber` objects specifying the operator type (see `NSPredicateOperatorType` for possible values).

options
Options for the predicate (see `NSComparisonPredicate_Options` for possible values).

Return Value
A row template of the “pop-up-pop-up-pop-up”-form, with the left and right popups representing the left and right expression arrays leftExpressions and rightExpressions, and the center popup representing the operators.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

leftExpressions

Returns the left hand expressions for the receiver.

```
- (NSArray *)leftExpressions
```

Return Value
The left hand expressions for the receiver

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

matchForPredicate:

Returns a positive number if the receiver can represent a given predicate, and `0` if it cannot.

```
- (double)matchForPredicate:(NSPredicate *)predicate
```

Return Value
A positive number if the template can represent *predicate*, and `0` if it cannot.

Discussion
By default, returns values in the range `0` to `1`.

The highest match among all the templates determines which template is responsible for displaying the predicate. You can override this to determine which predicates your custom template handles.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

modifier

Returns the comparison predicate modifier for the receiver.

```
- (NSComparisonPredicateModifier)modifier
```

Return Value
The comparison predicate modifier for the receiver.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

operators

Returns the array of operators for the receiver.

```
- (NSArray *)operators
```

Return Value
The array of operators for the receiver.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

options

Returns the comparison predicate options for the receiver.

```
- (NSUInteger)options
```

Return Value
The comparison predicate options for the receiver. See `NSComparisonPredicate_Options` for possible values. Returns `0` if this does not apply (for example, for a compound template initialized with `initWithCompoundTypes:` (page 2022)).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

predicateWithSubpredicates:

Returns the predicate represented by the receiver's views' values and the given sub-predicates.

```
- (NSPredicate *)predicateWithSubpredicates:(NSArray *)subpredicates
```

Parameters
subpredicates
An array of predicates.

Return Value
The predicate represented by the values of the template's views and the given subpredicates. You can override this method to return the predicate represented by your custom views.

Discussion
This method is only called if `matchForPredicate:` (page 2025) returned a positive value for the receiver.

You can override this method to return the predicate represented by a custom view.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

rightExpressionAttributeType

Returns the attribute type of the receiver's right expression.

```
- (NSAttributeType)rightExpressionAttributeType
```

Return Value
The attribute type of the receiver's right expression.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPredicateEditorRowTemplate.h

rightExpressions

Returns the right hand expressions for the receiver.

```
- (NSArray *)rightExpressions
```

Return Value
The right hand expressions for the receiver

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPredicateEditorRowTemplate.h

setPredicate:

Sets the value of the views according to the given predicate.

```
- (void)setPredicate:(NSPredicate *)predicate
```

Parameters
predicate
The predicate value for the receiver.

Discussion
This method is only called if `matchForPredicate:` (page 2025) returned a positive value for the receiver.

You can override this to set the values of custom views.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSPredicateEditorRowTemplate.h

templateViews

Returns the views for the receiver.

```
- (NSArray *)templateViews
```

Return Value
The views for the receiver.

Discussion
Instances of `NSPopUpButton` are treated specially by `NSPredicateEditor`; their menu items are merged into a single popup button, and matching menu item titles are combined. In this way, a single tree is built from the separate templates.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPredicateEditorRowTemplate.h`

NSPrinter Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSPrinter.h
Companion guide	Printing Programming Topics for Cocoa

Overview

An `NSPrinter` object describes a printer's capabilities as defined in its PPD file. An `NSPrinter` object can be constructed by specifying either the printer name or the make and model of an available printer. You use a printer object to get information about printers, not to modify printer attributes or control a printing job.

Adopted Protocols

NSCoding
- `- encodeWithCoder:`
- `- initWithCoder:`

NSCopying
- `- copyWithZone:`

Tasks

Creating an NSPrinter

`+ printerWithName:` (page 2032)
Creates and returns an `NSPrinter` object initialized with the specified printer name.

+ `printerWithType:` (page 2033)
Creates and returns an `NSPrinter` object initialized to the first available printer with the specified make and model information.

Getting General Printer Information

+ `printerNames` (page 2031)
Returns the names of all available printers.

+ `printerTypes` (page 2032)
Returns descriptions of the makes and models of all available printers.

Getting Attributes

- `name` (page 2038)
Returns the printer's name.

- `type` (page 2041)
Returns a description of the printer's make and model.

Getting Specific Information

- `pageSizeForPaper:` (page 2038)
Returns the size of the page for the specified paper type.

- `languageLevel` (page 2037)
Returns the PostScript language level recognized by the printer.

Querying the Tables

- `isKey:inTable:` (page 2037)
Returns a Boolean value indicating whether the specified key is in the specified table.

- `stringForKey:inTable:` (page 2040)
Returns the first occurrence of a value associated with specified key.

- `stringListForKey:inTable:` (page 2041)
Returns an array of strings, one for each occurrence, associated with specified key.

- `booleanForKey:inTable:` (page 2034)
Returns the Boolean value associated with the specified key.

- `floatForKey:inTable:` (page 2035)
Returns the floating-point value associated with the specified key.

- `intForKey:inTable:` (page 2036)
Returns the integer value associated with the specified key.

- `rectForKey:inTable:` (page 2039)
Returns the rectangle associated with the specified key.

- `sizeForKey:inTable:` (page 2039)
Returns the size data type associated with the specified key.

- `statusForTable:` (page 2040)
  Returns the status of the specified table.
- `deviceDescription` (page 2034)
  Returns a dictionary of keys and values describing the device.

Deprecated Methods

+ `printerWithName:domain:includeUnavailable:` (page 2032) Deprecated in Mac OS X v10.2
  Deprecated.
- `acceptsBinary` (page 2033) Deprecated in Mac OS X v10.2
  Deprecated.
- `domain` (page 2034) Deprecated in Mac OS X v10.2
  Deprecated.
- `host` (page 2035) Deprecated in Mac OS X v10.2
  Deprecated.
- `imageRectForPaper:` (page 2035) Deprecated in Mac OS X v10.2
  Deprecated.
- `isColor` (page 2036) Deprecated in Mac OS X v10.2
  Deprecated.
- `isFontAvailable:` (page 2036) Deprecated in Mac OS X v10.2
  Deprecated.
- `isOutputStackInReverseOrder` (page 2037) Deprecated in Mac OS X v10.2
  Deprecated.
- `note` (page 2038) Deprecated in Mac OS X v10.2
  Deprecated.

Class Methods

printerNames

Returns the names of all available printers.

```
+ (NSArray *)printerNames
```

Return Value
An array of `NSString` objects, each of which contains the name of an available printer.

Discussion
The user constructs the list of available printers using the Print Center application.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ `printerTypes` (page 2032)

- name (page 2038)

Declared In
NSPrinter.h

printerTypes

Returns descriptions of the makes and models of all available printers.

```
+ (NSArray *)printerTypes
```

Return Value
An array of NSString objects, each of which contains the make and model information for a supported printer.

Availability
Available in Mac OS X v10.0 and later.

See Also
- type (page 2041)

Declared In
NSPrinter.h

printerWithName:

Creates and returns an NSPrinter object initialized with the specified printer name.

```
+ (NSPrinter *)printerWithName:(NSString *)name
```

Parameters

name
The name of the printer.

Return Value
An initialized NSPrinter object, or nil if the specified printer was not available.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ printerWithType: (page 2033)
+ printerNames (page 2031)
- name (page 2038)

Declared In
NSPrinter.h

printerWithName:domain:includeUnavailable:

Deprecated. (Deprecated in Mac OS X v10.2.)

```
+ (NSPrinter *)printerWithName:(NSString *)name domain:(NSString *)domain
    includeUnavailable:(BOOL)includeUnavailable
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSPrinter.h

printerWithType:

Creates and returns an `NSPrinter` object initialized to the first available printer with the specified make and model information.

```
+ (NSPrinter *)printerWithType:(NSString *)type
```

Parameters

type

A string describing the make and model information. You can get this string using the `printerTypes` (page 2032) method.

Return Value
An initialized `NSPrinter` object, or `nil` if the specified printer was not available.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ printerWithName:` (page 2032)
`- type` (page 2041)

Declared In
NSPrinter.h

Instance Methods

acceptsBinary

Deprecated. (Deprecated in Mac OS X v10.2.)

```
- (BOOL)acceptsBinary
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSPrinter.h

booleanForKey:inTable:

Returns the Boolean value associated with the specified key.

```
- (BOOL)booleanForKey:(NSString *)key inTable:(NSString *)table
```

Parameters

key

The key whose value you want.

table

The name of a table from the printer's PPD file.

Return Value

The Boolean value associated with the key. Returns `NO` if the key is not in the table or the receiver lacks a PPD file.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `isKey:inTable:` (page 2037)
- `stringForKey:inTable:` (page 2040)

Declared In

`NSPrinter.h`

deviceDescription

Returns a dictionary of keys and values describing the device.

```
- (NSDictionary *)deviceDescription
```

Return Value

A dictionary of the device properties. See `NSGraphics.h` for possible keys. The only key guaranteed to exist is `NSDeviceIsPrinter`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSPrinter.h`

domain

Deprecated. (Deprecated in Mac OS X v10.2.)

```
- (NSString *)domain
```

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.2.

Declared In
NSPrinter.h

floatForKey:inTable:

Returns the floating-point value associated with the specified key.

```
- (float)floatForKey:(NSString *)key inTable:(NSString *)table
```

Parameters

key
The key whose value you want.

table
The name of a table from the printer's PPD file.

Return Value
The floating-point value. Returns `0.0` if the key is not in the table or the receiver lacks a PPD file.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isKey:inTable:` (page 2037)
- `stringForKey:inTable:` (page 2040)

Declared In
NSPrinter.h

host

Deprecated. (Deprecated in Mac OS X v10.2.)

```
- (NSString *)host
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSPrinter.h

imageRectForPaper:

Deprecated. (Deprecated in Mac OS X v10.2.)

```
- (NSRect)imageRectForPaper:(NSString *)paperName
```

Discussion
If used, it attempts to determine and return the bounds of the imagable area for a particular paper named *paperName*, but the result is not completely reliable.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

See Also
- `pageSizeForPaper:` (page 2038)

Declared In
`NSPrinter.h`

intForKey:inTable:

Returns the integer value associated with the specified key.

```
- (int)intForKey:(NSString *)key inTable:(NSString *)table
```

Parameters

key
The key whose value you want.

table
The name of a table from the printer's PPD file.

Return Value
The integer value. Returns `0` if the key is not in the table or the receiver lacks a PPD file.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isKey:inTable:` (page 2037)
- `stringForKey:inTable:` (page 2040)

Declared In
`NSPrinter.h`

isColor

Deprecated. (Deprecated in Mac OS X v10.2.)

```
- (BOOL)isColor
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
`NSPrinter.h`

isFontAvailable:

Deprecated. (Deprecated in Mac OS X v10.2.)

```
- (BOOL)isFontAvailable:(NSString *)faceName
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSPrinter.h

isKey:inTable:

Returns a Boolean value indicating whether the specified key is in the specified table.

```
- (BOOL)isKey:(NSString *)key inTable:(NSString *)table
```

Parameters

key
The key whose value you want.

table
The name of a table from the printer's PPD file.

Return Value
YES if the key is in the table; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPrinter.h

isOutputStackInReverseOrder

Deprecated. (Deprecated in Mac OS X v10.2.)

```
- (BOOL)isOutputStackInReverseOrder
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSPrinter.h

languageLevel

Returns the PostScript language level recognized by the printer.

```
- (NSInteger)languageLevel
```

Return Value
The PostScript language level. The value is 0 if the receiver is not a PostScript printer.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPrinter.h

name

Returns the printer's name.

```
- (NSString *)name
```

Return Value
The printer name.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ printerNames (page 2031)
+ printerWithName: (page 2032)

Declared In
NSPrinter.h

note

Deprecated. (Deprecated in Mac OS X v10.2.)

```
- (NSString *)note
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSPrinter.h

pageSizeForPaper:

Returns the size of the page for the specified paper type.

```
- (NSSize)pageSizeForPaper:(NSString *)paperName
```

Parameters
paperName
Possible values are printer-dependent and are contained in the printer's PPD file. Typical values are "Letter" and "Legal".

Return Value
The size of the page, measured in points in the user coordinate space. The returned size is zero if the specified paper name is not recognized or its entry in the PPD file cannot be parsed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `imageRectForPaper:` (page 2035)

Declared In
`NSPrinter.h`

rectForKey:inTable:

Returns the rectangle associated with the specified key.

```
- (NSRect)rectForKey:(NSString *)key inTable:(NSString *)table
```

Parameters

key
The key whose value you want.

table
The name of a table from the printer's PPD file.

Return Value
The rectangle value. Returns `NSZeroRect` if the key is not in the table or the receiver lacks a PPD file.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isKey:inTable:` (page 2037)
- `stringForKey:inTable:` (page 2040)

Declared In
`NSPrinter.h`

sizeForKey:inTable:

Returns the size data type associated with the specified key.

```
- (NSSize)sizeForKey:(NSString *)key inTable:(NSString *)table
```

Parameters

key
The key whose value you want.

table
The name of a table from the printer's PPD file.

Return Value
The size value. Returns `NSZeroSize` if the key is not in the table or the receiver lacks a PPD file.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isKey:inTable:` (page 2037)
- `stringForKey:inTable:` (page 2040)

Declared In
`NSPrinter.h`

statusForTable:

Returns the status of the specified table.

```
- (NSPrinterTableStatus)statusForTable:(NSString *)table
```

Parameters

table
The name of a table from the printer's PPD file.

Return Value
One of the return values described in "Constants" (page 2042).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPrinter.h`

stringForKey:inTable:

Returns the first occurrence of a value associated with specified key.

```
- (NSString *)stringForKey:(NSString *)key inTable:(NSString *)table
```

Parameters

key
The key whose value you want.

table
The name of a table from the printer's PPD file.

Return Value
The value for the specified key, or `nil` if the key is not in the table. The returned string may also be empty.

Discussion
If *key* is a main keyword only, and if that keyword has options in the PPD file, this method returns an empty string. Use `stringListForKey:inTable:` (page 2041) to retrieve the values for all occurrences of a main keyword.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isKey:inTable:` (page 2037)
- `booleanForKey:inTable:` (page 2034)
- `floatForKey:inTable:` (page 2035)

- intForKey:inTable: (page 2036)
- rectForKey:inTable: (page 2039)
- sizeForKey:inTable: (page 2039)

Declared In
NSPrinter.h

stringListForKey:inTable:

Returns an array of strings, one for each occurrence, associated with specified key.

```
- (NSArray *)stringListForKey:(NSString *)key inTable:(NSString *)table
```

Parameters

key
The key whose value you want.

table
The name of a table from the printer's PPD file.

Return Value
An array of NSString objects, each containing a value associated with the specified key. Returns nil if the key is not in the table.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isKey:inTable: (page 2037)
- stringForKey:inTable: (page 2040)

Declared In
NSPrinter.h

type

Returns a description of the printer's make and model.

```
- (NSString *)type
```

Return Value
A description of the printer's make and model.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ printerTypes (page 2032)

Declared In
NSPrinter.h

Constants

NSPrinterTableStatus

These constants describe the state of a printer information table stored by an `NSPrinter` object.

```
typedef enum _NSPrinterTableStatus {
    NSPrinterTableOK = 0,
    NSPrinterTableNotFound = 1,
    NSPrinterTableError = 2
} NSPrinterTableStatus;
```

Constants

`NSPrinterTableOK`
Printer table was found and is valid.
Available in Mac OS X v10.0 and later.
Declared in `NSPrinter.h`.

`NSPrinterTableNotFound`
Printer table was not found.
Available in Mac OS X v10.0 and later.
Declared in `NSPrinter.h`.

`NSPrinterTableError`
Printer table is not valid.
Available in Mac OS X v10.0 and later.
Declared in `NSPrinter.h`.

Discussion

These constants are used by `statusForTable:` (page 2040)..

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSPrinter.h`

NSPrintInfo Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSPrintInfo.h
Companion guide	Printing Programming Topics for Cocoa
Related sample code	ImageApp Quartz Composer WWDC 2005 TextEdit QuickLookSketch Sketch+Accessibility Sketch-112

Overview

An `NSPrintInfo` object stores information that's used to generate printed output. A shared `NSPrintInfo` object is automatically created for an application and is used by default for all printing jobs for that application.

The printing information in an `NSPrintInfo` object is stored in a dictionary. To access the standard attributes in the dictionary directly, this class defines a set of keys and provides the `dictionary` (page 2049) method. You can also initialize an instance of this class using the `initWithDictionary:` (page 2050) method.

You can use this dictionary to store custom information associated with a print job. Any non-object values should be stored as `NSNumber` or `NSValue` objects in the dictionary. See *NSNumber Class Reference* for a list of types which should be stored as numbers. For other non-object values, use the `NSValue` class.

Beginning with Mac OS X v10.5, to store custom information that belongs in printing presets you should use the dictionary returned by the `printSettings` (page 2056) method.

Tasks

Initializing an NSPrintInfo

- `initWithDictionary:` (page 2050)
  Returns an `NSPrintInfo` object initialized with the parameters in the specified dictionary.

Managing the Shared NSPrintInfo

+ `setSharedPrintInfo:` (page 2047)
  Sets the shared `NSPrintInfo` object to the specified object.
+ `sharedPrintInfo` (page 2048)
  Returns the shared `NSPrintInfo` object.

Managing the Printing Rectangle

- `bottomMargin` (page 2049)
  Returns the height of the bottom margin.
- `imageablePageBounds` (page 2050)
  Returns the imageable area of a sheet of paper specified by the receiver.
- `leftMargin` (page 2052)
  Returns the width of the left margin.
- `orientation` (page 2053)
  Returns the orientation attribute.
- `paperName` (page 2054)
  Returns the name of the currently selected paper size.
- `localizedPaperName` (page 2053)
  Returns the human-readable name of the currently selected paper size, suitable for presentation in user interfaces.
- `paperSize` (page 2054)
  Returns the size of the paper.
- `rightMargin` (page 2057)
  Returns the width of the right margin.
- `setBottomMargin:` (page 2058)
  Sets the bottom margin to the specified size.
- `setLeftMargin:` (page 2060)
  Sets the left margin to the specified size.
- `setOrientation:` (page 2060)
  Sets the page orientation to the specified value.
- `setPaperName:` (page 2061)
  Sets the paper name to the specified value.

- `setPaperSize:` (page 2061)
  Sets the width and height of the paper to the specified size.
- `setRightMargin:` (page 2062)
  Sets the right margin to the specified size.
- `setTopMargin:` (page 2064)
  Sets the top margin to the specified size.
- `topMargin` (page 2065)
  Returns the top margin.

Pagination

- `horizontalPagination` (page 2050)
  Returns the horizontal pagination mode.
- `setHorizontalPagination:` (page 2059)
  Sets the horizontal pagination to the specified mode.
- `setVerticalPagination:` (page 2065)
  Sets the vertical pagination to the specified mode.
- `verticalPagination` (page 2067)
  Returns the vertical pagination mode.

Positioning the Image on the Page

- `isHorizontallyCentered` (page 2051)
  Returns a Boolean value indicating whether the image is centered horizontally.
- `isVerticallyCentered` (page 2052)
  Returns a Boolean value indicating whether the image is centered vertically.
- `setHorizontallyCentered:` (page 2058)
  Sets whether the image is centered horizontally.
- `setVerticallyCentered:` (page 2064)
  Sets whether the image is centered vertically.

Specifying the Printer

- `printer` (page 2056)
  Returns the `NSPrinter` object to be used for printing.
- `setPrinter:` (page 2062)
  Sets the printer object used for subsequent printing jobs.

Controlling Printing

- `jobDisposition` (page 2052)
  Returns the action specified for the job.

- `setJobDisposition:` (page 2059)
  Sets the action specified for the job
- `setUpPrintOperationDefaultValues` (page 2064)
  Validates the attributes encapsulated by the receiver.

Accessing the Print Info Dictionary

- `dictionary` (page 2049)
  Returns the receiver's dictionary that contains the printing attributes.

Print Settings Convenience Methods

- `isSelectionOnly` (page 2051)
  Returns whether only the currently selected contents should be printed.
- `scalingFactor` (page 2058)
  Returns the current scaling factor.
- `setScalingFactor:` (page 2063)
  Sets the print info's scaling factor.
- `setSelectionOnly:` (page 2063)
  Sets whether only the current selection should be printed.

Accessing Core Printing Information

- `printSettings` (page 2056)
  Returns a mutable dictionary containing the print settings from Core Printing.
- `PMPrintSession` (page 2055)
  Returns a Core Printing object configured with the receiver's session information.
- `PMPageFormat` (page 2055)
  Returns a Core Printing object configured with the receiver's page format information.
- `PMPrintSettings` (page 2056)
  Returns a Core Printing object configured with the receiver's print settings information
- `updateFromPMPageFormat` (page 2066)
  Synchronizes the receiver's page format information with information from its associated `PMPageFormat` object.
- `updateFromPMPrintSettings` (page 2066)
  Synchronizes the receiver's print settings information with information from its associated `PMPrintSettings` object.

Deprecated Methods

+ `defaultPrinter` (page 2047)
  Deprecated.

Class Methods

defaultPrinter

Deprecated.

```
+ (NSPrinter *)defaultPrinter
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPrintInfo.h

setDefaultPrinter:

Deprecated. (Deprecated in Mac OS X v10.2.)

```
+ (void)setDefaultPrinter:(NSPrinter *)aPrinter
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSPrintInfo.h

setSharedPrintInfo:

Sets the shared NSPrintInfo object to the specified object.

```
+ (void)setSharedPrintInfo:(NSPrintInfo *)printInfo
```

Parameters
printInfo
The new shared printer information. This value must not be nil.

Discussion
The shared NSPrintInfo object defines the settings for the NSPageLayout panel and print operations that will be used if no NSPrintInfo object is specified for those operations.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ sharedPrintInfo (page 2048)

Related Sample Code
From A View to A Movie
From A View to A Picture
GLSL Showpiece Lite
GLUT

Declared In
NSPrintInfo.h

sharedPrintInfo

Returns the shared NSPrintInfo object.

```
+ (NSPrintInfo *)sharedPrintInfo
```

Return Value
The shared printer information.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ setSharedPrintInfo: (page 2047)

Related Sample Code
From A View to A Movie
From A View to A Picture
GLSL Showpiece Lite
GLUT
Quartz Composer WWDC 2005 TextEdit

Declared In
NSPrintInfo.h

sizeForPaperName:

Deprecated. (Deprecated in Mac OS X v10.2.)

```
+ (NSSize)sizeForPaperName:(NSString *)name
```

Discussion
Use the pageSizeForPaper: (page 2038) method of NSPrinter instead.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.2.

Declared In
NSPrintInfo.h

Instance Methods

bottomMargin

Returns the height of the bottom margin.

```
- (CGFloat)bottomMargin
```

Return Value
The bottom margin, measured in points in the user coordinate space.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBottomMargin:` (page 2058)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSPrintInfo.h`

dictionary

Returns the receiver's dictionary that contains the printing attributes.

```
- (NSMutableDictionary *)dictionary
```

Discussion
The key-value pairs contained in the dictionary are described in "Constants" (page 2067). Modifying the returned dictionary changes the receiver's attributes.

This dictionary is key-value observing compliant.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSPrintInfo.h`

horizontalPagination

Returns the horizontal pagination mode.

```
- (NSPrintingPaginationMode)horizontalPagination
```

Return Value
One of the pagination modes described in "NSPrintingPaginationMode" (page 2070).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setVerticalPagination:` (page 2065)
- `verticalPagination` (page 2067)

Declared In
`NSPrintInfo.h`

imageablePageBounds

Returns the imageable area of a sheet of paper specified by the receiver.

```
- (NSRect)imageablePageBounds
```

Return Value
The imageable area, measured in points in the user coordinate space.

Discussion
This method takes into account the current printer, paper size, and orientation settings, but not scaling factors. "Imageable area" is the maximum area that can possibly be marked on by the printer hardware, not the area defined by the current margin settings.

The origin (0, 0) of the returned rectangle is in the lower-left corner of the oriented sheet. The imageable bounds may extend past the edges of the sheet when, for example, a printer driver specifies it so that borderless printing can be done reliably.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
ImageApp

Declared In
`NSPrintInfo.h`

initWithDictionary:

Returns an `NSPrintInfo` object initialized with the parameters in the specified dictionary.

```
- (id)initWithDictionary:(NSDictionary *)aDictionary
```

Parameters

aDictionary

The possible key-value pairs contained in *aDictionary* are described in "Constants" (page 2067).

Return Value

An initialized `NSPrintInfo` object, or nil if the object could not be created.

Discussion

This method is the designated initializer for this class. Non-object values should be stored in `NSValue` objects (or an appropriate subclass like `NSNumber`) in the dictionary. See `NSNumber` for a list of types which should be stored using the `NSNumber` class; otherwise use `NSValue`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- dictionary` (page 2049)

Declared In

`NSPrintInfo.h`

isHorizontallyCentered

Returns a Boolean value indicating whether the image is centered horizontally.

```
- (BOOL)isHorizontallyCentered
```

Return Value

`YES` if the image is centered horizontally; otherwise, `NO`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- isVerticallyCentered` (page 2052)

`- setHorizontallyCentered:` (page 2058)

Declared In

`NSPrintInfo.h`

isSelectionOnly

Returns whether only the currently selected contents should be printed.

```
- (BOOL)isSelectionOnly
```

Return Value

`YES` if only the currently selected contents should be printed, otherwise `NO`.

Discussion

This method is key-value observing compliant.

Availability

Available in Mac OS X v10.6 and later.

See Also
- `setSelectionOnly:` (page 2063)

Declared In
`NSPrintInfo.h`

isVerticallyCentered

Returns a Boolean value indicating whether the image is centered vertically.

```
- (BOOL)isVerticallyCentered
```

Return Value
`YES` if the image is centered vertically; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isHorizontallyCentered` (page 2051)
- `setVerticallyCentered:` (page 2064)

Declared In
`NSPrintInfo.h`

jobDisposition

Returns the action specified for the job.

```
- (NSString *)jobDisposition
```

Return Value
One of the following value:

- `NSPrintSpoolJob` is a normal print job.
- `NSPrintPreviewJob` sends the print job to the Preview application.
- `NSPrintSaveJob` saves the print job to a file.
- `NSPrintCancelJob` aborts the print job.
- `NSPrintFaxJob` is deprecated.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPrintInfo.h`

leftMargin

Returns the width of the left margin.

```
- (CGFloat)leftMargin
```

Return Value
The left margin, measured in points in the user coordinate space.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setLeftMargin:` (page 2060)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSPrintInfo.h`

localizedPaperName

Returns the human-readable name of the currently selected paper size, suitable for presentation in user interfaces.

```
- (NSString *)localizedPaperName
```

Return Value
The name of the paper size.

Discussion
This is typically different from the name returned by `paperName` (page 2054), which is almost never suitable for presentation to the user.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSPrintInfo.h`

orientation

Returns the orientation attribute.

```
- (NSPrintingOrientation)orientation
```

Return Value
One of the following values: `NSPortraitOrientation` or `NSLandscapeOrientation`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setOrientation:` (page 2060)

Declared In
`NSPrintInfo.h`

paperName

Returns the name of the currently selected paper size.

```
- (NSString *)paperName
```

Return Value
The string contains a value such as Letter or Legal. Paper names are implementation specific.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPaperName:` (page 2061)
- `localizedPaperName` (page 2053)

Declared In
`NSPrintInfo.h`

paperSize

Returns the size of the paper.

```
- (NSSize)paperSize
```

Return Value
The size of the paper, measured in points in the user coordinate space.

Discussion
This method is key-value observing compliant.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPaperSize:` (page 2061)

Related Sample Code
ImageApp
Quartz Composer WWDC 2005 TextEdit
Quartz2DBasics
Sketch+Accessibility
Sketch-112

Declared In
`NSPrintInfo.h`

PMPageFormat

Returns a Core Printing object configured with the receiver's page format information.

```
- (void *)PMPageFormat
```

Return Value
A pointer to a `PMPageFormat` object, an opaque data type that stores information such as the paper size, orientation, and scale of pages in a printing session. You should not call `PMRelease` to release the returned object, except to balance calls to `PMRetain` that your code also issued.

Discussion
The information in the returned `PMPageFormat` object is consistent with the receiver's page format information at the time this method is called. Subsequent changes to the receiving `NSPrintInfo` object do not result in changes to the information in the `PMPageFormat` object.

If you make changes to the data in the `PMPageFormat` object, you should invoke the `updateFromPMPageFormat` method to synchronize those changes with the `NSPrintInfo` object that created the object.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `updateFromPMPageFormat` (page 2066)

Declared In
`NSPrintInfo.h`

PMPrintSession

Returns a Core Printing object configured with the receiver's session information.

```
- (void *)PMPrintSession
```

Return Value
A pointer to a `PMPrintSession` object, an opaque type that stores information about a print job. You should not call `PMRelease` to release the returned object, except to balance calls to `PMRetain` that your code also issued.

Discussion
The information in the returned `PMPrintSession` object is consistent with the receiver's session information at the time this method is called. Subsequent changes to the receiving `NSPrintInfo` object do not result in changes to the information in the `PMPrintSession` object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPrintInfo.h`

PMPrintSettings

Returns a Core Printing object configured with the receiver's print settings information

```
- (void *)PMPrintSettings
```

Return Value
A pointer to a `PMPrintSettings` object, an opaque data type used to store information such as the number of copies and the range of pages in a printing session. You should not call `PMRelease` to release the returned object, except to balance calls to `PMRetain` that your code also issued.

Discussion
The information in the returned `PMPrintSettings` object is consistent with the receiver's print settings at the time this method is called. Subsequent changes to the receiving `NSPrintInfo` object do not result in changes to the information in the `PMPrintSettings` data type.

If you make changes to the data in the `PMPrintSettings` object, you should invoke the `updateFromPMPrintSettings` method to synchronize those changes with the `NSPrintInfo` object that created the object.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `updateFromPMPrintSettings` (page 2066)

Declared In
`NSPrintInfo.h`

printer

Returns the `NSPrinter` object to be used for printing.

```
- (NSPrinter *)printer
```

Return Value
The printer object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPrinter:` (page 2062)

Declared In
`NSPrintInfo.h`

printSettings

Returns a mutable dictionary containing the print settings from Core Printing.

```
- (NSMutableDictionary *)printSettings
```

Return Value
A mutable dictionary containing the printing system's current settings.

Discussion
You can use this method to get and set values from the system print settings. The keys in the returned dictionary represent the values returned by the Core Printing function `PMPrintSettingsGetValue`. They correspond to the settings currently in the print panel and include everything from custom values set by your accessory panels to values provided by the printer driver's print dialog extension.

Adding keys to the dictionary is equivalent to calling the Core Printing function `PMPrintSettingsSetValue`. Your new keys are added to the current print settings and are saved with any user preset files generated by the Mac OS X printing system. Because the print settings are stored in a property list, any values you add to the dictionary must correspond to scalar types such as strings, numbers, dates, booleans, and data objects or collection types such as dictionaries and arrays.

Other parts of the printing system use key strings like `com.apple.print.PrintSettings.PMColorSyncProfileID` to identify print settings. Cocoa replaces the periods in such strings with underscores. Thus, the preceding key string would be `com_apple_print_PrintSettings_PMColorSyncProfileID` instead. If you use reverse-DNS style key strings for your custom attributes, you should follow the same convention of using underscore characters instead of periods.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPrintInfo.h`

rightMargin

Returns the width of the right margin.

```
- (CGFloat)rightMargin
```

Return Value
The right margin, measured in points in the user coordinate space.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setRightMargin:` (page 2062)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSPrintInfo.h`

scalingFactor

Returns the current scaling factor.

```
- (CGFloat)scalingFactor
```

Return Value
The current scaling factor.

Discussion
This method is key-value observing compliant.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setScalingFactor: (page 2063)

Declared In
NSPrintInfo.h

setBottomMargin:

Sets the bottom margin to the specified size.

```
- (void)setBottomMargin:(CGFloat)margin
```

Parameters
margin
The new size for the right margin, measured in points in the user coordinate space.

Availability
Available in Mac OS X v10.0 and later.

See Also
- bottomMargin (page 2049)

Related Sample Code
From A View to A Movie
From A View to A Picture
GLSL Showpiece Lite
ImageApp
Quartz Composer WWDC 2005 TextEdit

Declared In
NSPrintInfo.h

setHorizontallyCentered:

Sets whether the image is centered horizontally.

```
- (void)setHorizontallyCentered:(BOOL)flag
```

Parameters

flag

YES if you want the image to be centered horizontally; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isHorizontallyCentered (page 2051)
- isVerticallyCentered (page 2052)
- setVerticallyCentered: (page 2064)

Declared In
NSPrintInfo.h

setHorizontalPagination:

Sets the horizontal pagination to the specified mode.

```
- (void)setHorizontalPagination:(NSPrintingPaginationMode)mode
```

Parameters

mode

One of the pagination modes described in "NSPrintingPaginationMode" (page 2070).

Availability
Available in Mac OS X v10.0 and later.

See Also
- horizontalPagination (page 2050)
- setVerticalPagination: (page 2065)
- verticalPagination (page 2067)

Related Sample Code
From A View to A Movie
From A View to A Picture
GLSL Showpiece Lite
GLUT

Declared In
NSPrintInfo.h

setJobDisposition:

Sets the action specified for the job

```
- (void)setJobDisposition:(NSString *)disposition
```

Parameters

disposition

One of the following value:

- `NSPrintSpoolJob` is a normal print job.
- `NSPrintPreviewJob` sends the print job to the Preview application.
- `NSPrintSaveJob` saves the print job to a file.
- `NSPrintCancelJob` aborts the print job.
- `NSPrintFaxJob` is deprecated.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- jobDisposition` (page 2052)

Declared In
`NSPrintInfo.h`

setLeftMargin:

Sets the left margin to the specified size.

```
- (void)setLeftMargin:(CGFloat)margin
```

Parameters

margin

The new size for the left margin, measured in points in the user coordinate space.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- leftMargin` (page 2052)

Related Sample Code
From A View to A Movie
From A View to A Picture
GLSL Showpiece Lite
ImageApp
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSPrintInfo.h`

setOrientation:

Sets the page orientation to the specified value.

```
- (void)setOrientation:(NSPrintingOrientation)orientation
```

Parameters

orientation

This printing orientation. See "NSPrintingOrientation" (page 2071) for possible values..

Discussion

For consistency, this method may change either the paper name or the paper size.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `dictionary` (page 2049)
- `initWithDictionary:` (page 2050)
- `orientation` (page 2053)

Related Sample Code

From A View to A Movie

From A View to A Picture

GLSL Showpiece Lite

Declared In

`NSPrintInfo.h`

setPaperName:

Sets the paper name to the specified value.

```
- (void)setPaperName:(NSString *)name
```

Parameters

name

The name for the paper size. The string contains a value such as Letter or Legal. Paper names are implementation specific.

Discussion

For consistency, this method may change either the paper size or the page orientation.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `dictionary` (page 2049)
- `initWithDictionary:` (page 2050)
- `paperName` (page 2054)

Declared In

`NSPrintInfo.h`

setPaperSize:

Sets the width and height of the paper to the specified size.

```
- (void)setPaperSize:(NSSize)aSize
```

Parameters

aSize

The new size of the paper, measured in points in the user coordinate space.

Discussion

For consistency, this method may change either the paper name or the page orientation.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `dictionary` (page 2049)
- `initWithDictionary:` (page 2050)
- `paperSize` (page 2054)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

`NSPrintInfo.h`

setPrinter:

Sets the printer object used for subsequent printing jobs.

```
- (void)setPrinter:(NSPrinter *)printer
```

Parameters

printer

The printer object.

Discussion

This method iterates through the receiver's dictionary. If a feature in the dictionary is not supported by the new printer (as determined by a query to the PPD file), that feature is removed from the dictionary.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `printer` (page 2056)

Declared In

`NSPrintInfo.h`

setRightMargin:

Sets the right margin to the specified size.

```
- (void)setRightMargin:(CGFloat)margin
```

Parameters

margin

The new size for the right margin, measured in points in the user coordinate space.

Availability
Available in Mac OS X v10.0 and later.

See Also
- rightMargin (page 2057)

Related Sample Code
From A View to A Movie
From A View to A Picture
GLSL Showpiece Lite
ImageApp
Quartz Composer WWDC 2005 TextEdit

Declared In
NSPrintInfo.h

setScalingFactor:

Sets the print info's scaling factor.

```
- (void)setScalingFactor:(CGFloat)scalingFactor
```

Parameters

scalingFactor

The new scaling factor.

Availability
Available in Mac OS X v10.6 and later.

See Also
- scalingFactor (page 2058)

Declared In
NSPrintInfo.h

setSelectionOnly:

Sets whether only the current selection should be printed.

```
- (void)setSelectionOnly:(BOOL)selectionOnly
```

Parameters

selectionOnly

YES if only the current selection should be printed, otherwise NO.

Availability
Available in Mac OS X v10.6 and later.

See Also
- isSelectionOnly (page 2051)

Declared In
NSPrintInfo.h

setTopMargin:

Sets the top margin to the specified size.

```
- (void)setTopMargin:(CGFloat)margin
```

Parameters

margin
The new size for the top margin, measured in points in the user coordinate space.

Availability
Available in Mac OS X v10.0 and later.

See Also
- topMargin (page 2065)

Related Sample Code
From A View to A Movie
From A View to A Picture
GLSL Showpiece Lite
ImageApp
Quartz Composer WWDC 2005 TextEdit

Declared In
NSPrintInfo.h

setUpPrintOperationDefaultValues

Validates the attributes encapsulated by the receiver.

```
- (void)setUpPrintOperationDefaultValues
```

Discussion
Invoked when the print operation is about to start. Subclasses may override this method to set default values for any attributes that are not set.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPrintInfo.h

setVerticallyCentered:

Sets whether the image is centered vertically.

```
- (void)setVerticallyCentered:(BOOL)flag
```

Parameters

flag

`YES` if you want the image to be centered vertically; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isHorizontallyCentered` (page 2051)
- `isVerticallyCentered` (page 2052)
- `setHorizontallyCentered:` (page 2058)

Declared In
`NSPrintInfo.h`

setVerticalPagination:

Sets the vertical pagination to the specified mode.

```
- (void)setVerticalPagination:(NSPrintingPaginationMode)mode
```

Parameters

mode

One of the pagination modes described in "NSPrintingPaginationMode" (page 2070).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `horizontalPagination` (page 2050)
- `setHorizontalPagination:` (page 2059)
- `verticalPagination` (page 2067)

Related Sample Code
From A View to A Movie
From A View to A Picture
GLSL Showpiece Lite
GLUT

Declared In
`NSPrintInfo.h`

topMargin

Returns the top margin.

```
- (CGFloat)topMargin
```

Return Value
The top margin, measured in points in the user coordinate space.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTopMargin:` (page 2064)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSPrintInfo.h`

updateFromPMPageFormat

Synchronizes the receiver's page format information with information from its associated `PMPageFormat` object.

```
- (void)updateFromPMPageFormat
```

Discussion
You should use this method after making changes to the `PMPageFormat` object obtained from the receiver. Each `NSPrintInfo` object keeps track of the object returned from its `PMPageFormat` method and obtains any updated information from the object directly. You only need to synchronize the objects once when you have made all of the desired changes.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `PMPageFormat` (page 2055)

Declared In
`NSPrintInfo.h`

updateFromPMPrintSettings

Synchronizes the receiver's print settings information with information from its associated `PMPrintSettings` object.

```
- (void)updateFromPMPrintSettings
```

Discussion
You should use this method after making changes to the `PMPrintSettings` object obtained from the receiver. Each `NSPrintInfo` object keeps track of the object returned from its `PMPrintSettings` method and obtains any updated information from the object directly. You only need to synchronize the objects once when you have made all of the desired changes.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `PMPrintSettings` (page 2056)

Declared In
`NSPrintInfo.h`

verticalPagination

Returns the vertical pagination mode.

```
- (NSPrintingPaginationMode)verticalPagination
```

Return Value
One of the pagination modes described in "Constants" (page 2067).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `horizontalPagination` (page 2050)
- `setHorizontalPagination:` (page 2059)

Declared In
`NSPrintInfo.h`

Constants

Print job attributes

These constants specify dictionary keys to access print job attributes.

```
NSString *const NSPrintPrinter;
NSString *const NSPrintCopies;
NSString *const NSPrintAllPages;
NSString *const NSPrintFirstPage;
NSString *const NSPrintLastPage;
NSString *const NSPrintMustCollate;
NSString *const NSPrintReversePageOrder;
NSString *const NSPrintJobDisposition;
NSString *const NSPrintSavePath;
NSString *const NSPrintPagesAcross;
NSString *const NSPrintPagesDown;
NSString *const NSPrintTime;
NSString *const NSPrintDetailedErrorReporting;
NSString *const NSPrintFaxNumber;
NSString *const NSPrintPrinterName;
NSString *const NSPrintHeaderAndFooter;
NSString *const NSPrintSelectionOnly;
NSString *const NSPrintJobSavingURL;
NSString *const NSPrintJobSavingFileNameExtensionHidden'
```

Constants

`NSPrintPrinter`

An `NSPrinter` object—the printer to use.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintCopies`

An `NSNumber` object containing an integer—the number of copies to spool.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintAllPages`

An `NSNumber` object containing a Boolean value—if `YES`, includes all pages in output.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintFirstPage`

An `NSNumber` object containing an integer value that specifies the first page in the print job.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintLastPage`

An `NSNumber` object containing an integer value that specifies the last page in the print job.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintMustCollate`

An `NSNumber` object containing a Boolean value—if `YES`, collates output.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintReversePageOrder`
An `NSNumber` object containing a Boolean value—if `YES`, prints first page last.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintJobDisposition`
An `NSString` object that specifies the job disposition.

`NSPrintSpoolJob`, `NSPrintPreviewJob`, `NSPrintSaveJob`, or `NSPrintCancelJob`. See `setJobDisposition:` (page 2059) for details.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintSavePath`
An `NSString` object that specifies the pathname to which the job file will be saved when the `jobDisposition` (page 2052) is `NSPrintSaveJob` (page 2071)..

Use `NSPrintJobSavingURL` (page 2070) instead.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintPagesAcross`
An `NSNumber` object that specifies the number of logical pages to be tiled horizontally on a physical sheet of paper.

Available in Mac OS X v10.4 and later.

Declared in `NSPrintInfo.h`.

`NSPrintPagesDown`
An `NSNumber` object that specifies the number of logical pages to be tiled vertically on a physical sheet of paper.

Available in Mac OS X v10.4 and later.

Declared in `NSPrintInfo.h`.

`NSPrintTime`
An `NSDate` object that specifies the time at which printing should begin.

Available in Mac OS X v10.4 and later.

Declared in `NSPrintInfo.h`.

`NSPrintDetailedErrorReporting`
An `NSNumber` object containing a Boolean value—if `YES`, produce detailed reports when an error occurs.

Available in Mac OS X v10.4 and later.

Declared in `NSPrintInfo.h`.

`NSPrintFaxNumber`
An `NSString` object that specifies a fax number.

Available in Mac OS X v10.4 and later.

Declared in `NSPrintInfo.h`.

`NSPrintPrinterName`
An `NSString` object that specifies the name of a printer.

Available in Mac OS X v10.4 and later.

Declared in `NSPrintInfo.h`.

`NSPrintHeaderAndFooter`
An `NSNumber` object containing a Boolean value—if `YES`, a standard header and footer are added outside the margins of each page.

Available in Mac OS X v10.4 and later.

Declared in `NSPrintInfo.h`.

`NSPrintSelectionOnly`
An `NSNumber` object containing a Boolean value—if `YES` only the current selection is printed.

Available in Mac OS X v10.6 and later.

Declared in `NSPrintInfo.h`.

`NSPrintJobSavingURL`
An `NSURL` containing the location to which the job file will be saved when the `jobDisposition` (page 2052) is `NSPrintSaveJob` (page 2071).

Available in Mac OS X v10.6 and later.

Declared in `NSPrintInfo.h`.

`NSPrintJobSavingFileNameExtensionHidden`
A boolean `NSNumber` indicating whether the job's file name extension should be hidden when the `jobDisposition` (page 2052) is `NSPrintSaveJob` (page 2071). The default is `NO`.

Available in Mac OS X v10.6 and later.

Declared in `NSPrintInfo.h`.

NSPrintingPaginationMode

These constants specify the different ways in which an image is divided into pages. They're used by `horizontalPagination` (page 2050), `setHorizontalPagination:` (page 2059), `verticalPagination` (page 2067), and `setVerticalPagination:` (page 2065).

```
enum {
    NSAutoPagination = 0,
    NSFitPagination  = 1,
    NSClipPagination = 2
};
typedef NSUInteger NSPrintingPaginationMode;
```

Constants

`NSAutoPagination`
The image is divided into equal-sized rectangles and placed in one column of pages.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSFitPagination`
The image is scaled to produce one column or one row of pages.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSClipPagination`
The image is clipped to produce one column or row of pages.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

NSPrintingOrientation

These constants specify page orientations used by the methods `orientation` (page 2053) and `setOrientation:` (page 2060).

```
enum {
    NSPortraitOrientation  = 0,
    NSLandscapeOrientation = 1
};
typedef NSUInteger NSPrintingOrientation;
```

Constants

`NSPortraitOrientation`
Orientation is portrait (page is taller than it is wide).
Available in Mac OS X v10.0 and later.
Declared in `NSPrintInfo.h`.

`NSLandscapeOrientation`
Orientation is landscape (page is wider than it is tall).
Available in Mac OS X v10.0 and later.
Declared in `NSPrintInfo.h`.

Print job dispositions

These constants specify valid values for the print job attribute `NSPrintJobDisposition` (page 2069). These constants are used by the `jobDisposition` (page 2052) and `setJobDisposition:` (page 2059) methods.

```
NSString *const NSPrintSpoolJob;
NSString *const NSPrintPreviewJob;
NSString *const NSPrintSaveJob;
NSString *const NSPrintCancelJob;
```

Constants

`NSPrintSpoolJob`
Normal print job.
Available in Mac OS X v10.0 and later.
Declared in `NSPrintInfo.h`.

`NSPrintPreviewJob`
Send to Preview application.
Available in Mac OS X v10.0 and later.
Declared in `NSPrintInfo.h`.

`NSPrintSaveJob`
Save to a file.
Available in Mac OS X v10.0 and later.
Declared in `NSPrintInfo.h`.

`NSPrintCancelJob`
Cancel print job.
Available in Mac OS X v10.0 and later.
Declared in `NSPrintInfo.h`.

Page setup attributes

These constants specify dictionary keys to access page format attributes.

```
NSString *NSPrintPaperName;
NSString *NSPrintPaperSize;
NSString *NSPrintOrientation;
NSString *NSPrintScalingFactor;
```

Constants

`NSPrintPaperName`

An `NSString` object containing the paper name.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintPaperSize`

An `NSSize` value specifying the height and width of paper in points.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintOrientation`

An `NSNumber` object containing an `NSPrintingOrientation`.

`NSPortraitOrientation` or `NSLandscapeOrientation`

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintScalingFactor`

Scale factor percentage before pagination.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

Declared In

`NSPrintInfo.h`

Pagination attributes

These constants specify dictionary keys to access pagination attributes.

```
NSString *NSPrintBottomMargin;
NSString *NSPrintHorizontalPagination;
NSString *NSPrintHorizontallyCentered;
NSString *NSPrintLeftMargin;
NSString *NSPrintRightMargin;
NSString *NSPrintTopMargin;
NSString *NSPrintVerticalPagination;
NSString *NSPrintVerticallyCentered;
```

Constants

`NSPrintLeftMargin`

`NSNumber`, containing a floating-point value that specifies the left margin, in points.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintRightMargin`
`NSNumber`, containing a floating-point value that specifies the right margin, in points.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintTopMargin`
`NSNumber`, containing a floating-point value that specifies the top margin, in points.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintBottomMargin`
`NSNumber`, containing a floating-point value that specifies the bottom margin, in points.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintHorizontallyCentered`
`NSNumber`, containing a Boolean value that is `YES` if pages are centered horizontally.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintVerticallyCentered`
`NSNumber`, containing a Boolean value that is `YES` if pages are centered vertically.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintHorizontalPagination`
`NSNumber`, containing a `NSPrintingPaginationMode` value.

`NSAutoPagination`, `NSFitPagination`, or `NSClipPagination`. See `setHorizontalPagination:` (page 2059) for details.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

`NSPrintVerticalPagination`
`NSNumber`, containing a `NSPrintingPaginationMode` value.

`NSAutoPagination`, `NSFitPagination`, or `NSClipPagination`. See `setVerticalPagination:` (page 2065) for details.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintInfo.h`.

Declared In
`NSPrintInfo.h`

Deprecated Printing Keys

These keys refer to older printing properties that are no longer used. (Deprecated. Use the keys described in "Print job attributes" (page 2067) instead.)

```
NSString *NSPrintFormName;
NSString *NSPrintJobFeatures;
NSString *NSPrintManualFeed;
NSString *NSPrintPagesPerSheet;
NSString *NSPrintPaperFeed;
NSString *NSPrintFaxReceiverNames;
NSString *NSPrintFaxReceiverNumbers;
NSString *NSPrintFaxSendTime;
NSString *NSPrintFaxUseCoverSheet;
NSString *NSPrintFaxCoverSheetName;
NSString *NSPrintFaxReturnReceipt;
NSString *NSPrintFaxHighResolution;
NSString *NSPrintFaxTrimPageEnds;
NSString *NSPrintFaxModem;
NSString *NSPrintFaxJob;
```

Constants

`NSPrintFormName`

Deprecated. Do not use.

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.2.

Declared in `NSPrintInfo.h`.

`NSPrintJobFeatures`

Deprecated. Do not use.

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.2.

Declared in `NSPrintInfo.h`.

`NSPrintManualFeed`

Deprecated. Do not use.

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.2.

Declared in `NSPrintInfo.h`.

`NSPrintPagesPerSheet`

Deprecated. Do not use.

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.2.

Declared in `NSPrintInfo.h`.

`NSPrintPaperFeed`

Deprecated. Do not use.

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.2.

Declared in `NSPrintInfo.h`.

`NSPrintFaxReceiverNames`

Deprecated. Do not use.

Available in Mac OS X v10.0 through Mac OS X v10.5.

Declared in `NSPrintInfo.h`.

`NSPrintFaxReceiverNumbers`
Deprecated. Do not use.
Available in Mac OS X v10.0 through Mac OS X v10.5.
Declared in `NSPrintInfo.h`.

`NSPrintFaxSendTime`
Deprecated. Do not use.
Available in Mac OS X v10.0 through Mac OS X v10.5.
Declared in `NSPrintInfo.h`.

`NSPrintFaxUseCoverSheet`
Deprecated. Do not use.
Available in Mac OS X v10.0 through Mac OS X v10.5.
Declared in `NSPrintInfo.h`.

`NSPrintFaxCoverSheetName`
Deprecated. Do not use.
Available in Mac OS X v10.0 through Mac OS X v10.5.
Declared in `NSPrintInfo.h`.

`NSPrintFaxReturnReceipt`
Deprecated. Do not use.
Available in Mac OS X v10.0 through Mac OS X v10.5.
Declared in `NSPrintInfo.h`.

`NSPrintFaxHighResolution`
Deprecated. Do not use.
Available in Mac OS X v10.0 through Mac OS X v10.5.
Declared in `NSPrintInfo.h`.

`NSPrintFaxTrimPageEnds`
Deprecated. Do not use.
Available in Mac OS X v10.0 through Mac OS X v10.5.
Declared in `NSPrintInfo.h`.

`NSPrintFaxModem`
Deprecated. Do not use.
Available in Mac OS X v10.0 through Mac OS X v10.5.
Declared in `NSPrintInfo.h`.

`NSPrintFaxJob`
Deprecated. Do not use.
Available in Mac OS X v10.0 through Mac OS X v10.5.
Declared in `NSPrintInfo.h`.

NSPrintOperation Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSPrintOperation.h
Companion guide	Printing Programming Topics for Cocoa
Related sample code	GLUT ImageApp Quartz Composer WWDC 2005 TextEdit Sketch+Accessibility Sketch-112

Overview

An `NSPrintOperation` object controls operations that generate Encapsulated PostScript (EPS) code, Portable Document Format (PDF) code, or print jobs. An `NSPrintOperation` object works in conjunction with two other objects: an `NSPrintInfo` object, which specifies how the code should be generated, and an `NSView` object, which generates the actual code.

It is important to note that the majority of methods in `NSPrintOperation` copy the instance of `NSPrintInfo` passed into them. Future changes to that print info are not reflected in the print info retained by the current `NSPrintOperation` object. All changes should be made to the print info before passing to the methods of this class. The only method in `NSPrintOperation` which does not copy the `NSPrintInfo` instance is `setPrintInfo:` (page 2097).

Note: You should not subclass `NSPrintOperation`. Methods that return an `NSPrintOperation` object return an instance of a concrete subclass whose implementation is private.

Tasks

Creating an NSPrintOperation

`+ EPSOperationWithView:insideRect:toData:` (page 2081)
Creates and returns a new `NSPrintOperation` object ready to control the copying of EPS graphics from the specified view.

`+ EPSOperationWithView:insideRect:toData:printInfo:` (page 2082)
Creates and returns a new `NSPrintOperation` object ready to control the copying of EPS graphics from the specified view using the specified print settings.

`+ EPSOperationWithView:insideRect:toPath:printInfo:` (page 2083)
Creates and returns a new `NSPrintOperation` object ready to control the copying of EPS graphics from the specified view and write the resulting data to the specified file.

`+ PDFOperationWithView:insideRect:toData:` (page 2083)
Creates and returns a new `NSPrintOperation` object ready to control the copying of PDF graphics from the specified view.

`+ PDFOperationWithView:insideRect:toData:printInfo:` (page 2084)
Creates and returns a new `NSPrintOperation` object ready to control the copying of PDF graphics from the specified view using the specified print settings.

`+ PDFOperationWithView:insideRect:toPath:printInfo:` (page 2085)
Creates and returns a new `NSPrintOperation` object ready to control the copying of PDF graphics from the specified view and write the resulting data to the specified file.

`+ printOperationWithView:` (page 2085)
Creates and returns an `NSPrintOperation` object ready to control the printing of the specified view.

`+ printOperationWithView:printInfo:` (page 2086)
Creates and returns an `NSPrintOperation` object ready to control the printing of the specified view using custom print settings.

Setting the Current NSPrintOperation for This Thread

`+ currentOperation` (page 2081)
Returns the current print operation for this thread.

`+ setCurrentOperation:` (page 2087)
Sets the current print operation for this thread.

Determining the Type of Operation

`- isCopyingOperation` (page 2090)
Returns a Boolean value indicating whether the receiver is an EPS or PDF copy operation.

Modifying the NSPrintInfo Object

- `printInfo` (page 2092)
  Returns the receiver's `NSPrintInfo` object.
- `setPrintInfo:` (page 2097)
  Sets the receiver's `NSPrintInfo` object.

Getting the NSView Object

- `view` (page 2101)
  Returns the view object that generates the actual data for the print operation.

Running a Print Operation

- `runOperation` (page 2093)
  Runs the print operation on the current thread.
- `runOperationModalForWindow:delegate:didRunSelector:contextInfo:` (page 2094)
  Runs the print operation, calling your custom delegate method upon completion.
- `cleanUpOperation` (page 2088)
  Called at the end of a print operation to remove the receiver as the current operation.
- `deliverResult` (page 2090)
  Delivers the results of the print operation to the intended destination.

Modifying the User Interface

- `showsPrintPanel` (page 2100)
  Returns a Boolean value indicating whether a print panel is displayed during the operation,
- `setShowsPrintPanel:` (page 2099)
  Sets whether the receiver displays a print panel for this operation.
- `showsProgressPanel` (page 2100)
  Returns a Boolean value indicating whether a progress panel is displayed during the operation.
- `setShowsProgressPanel:` (page 2099)
  Sets whether the receiver displays a progress panel for this operation.
- `jobTitle` (page 2091)
  Returns the title of the print job.
- `setJobTitle:` (page 2096)
  Assigns a custom title to the print job.
- `printPanel` (page 2093)
  Returns the `NSPrintPanel` object used when running the operation.
- `setPrintPanel:` (page 2097)
  Sets the `NSPrintPanel` object to be used during the operation.

Managing the Drawing Context

- `context` (page 2089)
  Returns the graphics context object used for generating output.
- `createContext` (page 2089)
  Creates the graphics context object used for drawing during the operation.
- `destroyContext` (page 2090)
  Destroys the receiver's graphics context.

Managing Page Information

- `currentPage` (page 2089)
  Returns the current page number being printed.
- `pageRange` (page 2092)
  Returns the range of pages associated with the print operation.
- `pageOrder` (page 2092)
  Returns the print order for the pages.
- `setPageOrder:` (page 2097)
  Sets the print order for the pages of the operation.

Managing Printing-Related Threads

- `canSpawnSeparateThread` (page 2088)
  Returns a Boolean value indicating whether the receiver is allowed to spawn a separate printing thread.
- `setCanSpawnSeparateThread:` (page 2095)
  Sets whether the receiver is allowed to spawn a separate printing thread.

Deprecated Methods

- `jobStyleHint` (page 2091)
  Returns the type of content that the print job is printing. (Deprecated. Use the `jobStyleHint` method of `NSPrintPanel` instead.)
- `setJobStyleHint:` (page 2096)
  Sets the type of content that the print job is printing. (Deprecated. Use the `setJobStyleHint:` method of `NSPrintPanel` instead.)
- `accessoryView` (page 2087)
  Returns the accessory view used by the receiver's print panel. (Deprecated. Use the `accessoryControllers` method of `NSPrintPanel` instead.)
- `setShowPanels:` (page 2098) Deprecated in Mac OS X v10.4 and later
  Sets whether the print operation should display a print panel. (Deprecated. Use `setShowsPrintPanel:` (page 2099) and `setShowsProgressPanel:` (page 2099) instead.)

- `showPanels` (page 2100) Deprecated in Mac OS X v10.4 and later
  Returns a Boolean value that indicates whether the print panel is to be displayed. (Deprecated. Use `showsPrintPanel` (page 2100) and `showsProgressPanel` (page 2100) instead.)
- `setAccessoryView:` (page 2095) Deprecated in Mac OS X v10.5
  Sets the custom accessory view to be displayed by the receiver's print panel. (Deprecated. Use the `addAccessoryController:` method of `NSPrintPanel` instead.)

Class Methods

currentOperation

Returns the current print operation for this thread.

```
+ (NSPrintOperation *)currentOperation
```

Return Value
The print operation object, or `nil` if there is no current operation.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ setCurrentOperation:` (page 2087)

Related Sample Code
Quartz2DBasics

Declared In
`NSPrintOperation.h`

EPSOperationWithView:insideRect:toData:

Creates and returns a new `NSPrintOperation` object ready to control the copying of EPS graphics from the specified view.

```
+ (NSPrintOperation *)EPSOperationWithView:(NSView *)aView insideRect:(NSRect)rect
    toData:(NSMutableData *)data
```

Parameters

aView
The view containing the data to be turned into EPS data.

rect
The portion of the view (specified in points in the view's coordinate space) to be rendered as EPS data.

data
An empty `NSMutableData` object. After the job is run, this object contains the EPS data.

Return Value
The new `NSPrintOperation` object. You must run the operation to generate the EPS data.

Discussion
The new `NSPrintOperation` object uses the default `NSPrintInfo` object. This method raises an `NSPrintOperationExistsException` if there is already a print operation in progress; otherwise the returned object is made the current print operation for this thread.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ EPSOperationWithView:insideRect:toData:printInfo:` (page 2082)
`+ EPSOperationWithView:insideRect:toPath:printInfo:` (page 2083)
`- runOperation` (page 2093)

Declared In
`NSPrintOperation.h`

EPSOperationWithView:insideRect:toData:printInfo:

Creates and returns a new `NSPrintOperation` object ready to control the copying of EPS graphics from the specified view using the specified print settings.

```
+ (NSPrintOperation *)EPSOperationWithView:(NSView *)aView insideRect:(NSRect)rect
    toData:(NSMutableData *)data printInfo:(NSPrintInfo *)aPrintInfo
```

Parameters

aView
The view containing the data to be turned into EPS data.

rect
The portion of the view (specified in points in the view's coordinate space) to be rendered as EPS data.

data
An empty `NSMutableData` object. After the job is run, this object contains the EPS data.

aPrintInfo
The print settings to use when generating the EPS data.

Return Value
The new `NSPrintOperation` object. You must run the operation to generate the EPS data.

Discussion
This method raises an `NSPrintOperationExistsException` if there is already a print operation in progress; otherwise the returned object is made the current print operation for this thread.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ EPSOperationWithView:insideRect:toData:` (page 2081)
`+ EPSOperationWithView:insideRect:toPath:printInfo:` (page 2083)
`- runOperation` (page 2093)

Declared In
`NSPrintOperation.h`

EPSOperationWithView:insideRect:toPath:printInfo:

Creates and returns a new `NSPrintOperation` object ready to control the copying of EPS graphics from the specified view and write the resulting data to the specified file.

```
+ (NSPrintOperation *)EPSOperationWithView:(NSView *)aView insideRect:(NSRect)rect
    toPath:(NSString *)path printInfo:(NSPrintInfo *)aPrintInfo
```

Parameters

aView

The view containing the data to be turned into EPS data.

rect

The portion of the view (specified in points in the view's coordinate space) to be rendered as EPS data.

path

The path to a file. After the job is run, this file contains the EPS data.

aPrintInfo

The print settings to use when generating the EPS data.

Return Value

The new `NSPrintOperation` object. You must run the operation to generate the EPS data.

Discussion

This method raises an `NSPrintOperationExistsException` if there is already a print operation in progress; otherwise the returned object is made the current print operation for this thread.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ EPSOperationWithView:insideRect:toData:` (page 2081)

`+ EPSOperationWithView:insideRect:toData:printInfo:` (page 2082)

`- runOperation` (page 2093)

Declared In

`NSPrintOperation.h`

PDFOperationWithView:insideRect:toData:

Creates and returns a new `NSPrintOperation` object ready to control the copying of PDF graphics from the specified view.

```
+ (NSPrintOperation *)PDFOperationWithView:(NSView *)aView insideRect:(NSRect)rect
    toData:(NSMutableData *)data
```

Parameters

aView

The view containing the data to be turned into PDF data.

rect

The portion of the view (specified in points in the view's coordinate space) to be rendered as PDF data.

data

An empty `NSMutableData` object. After the job is run, this object contains the PDF data.

Return Value
The new `NSPrintOperation` object. You must run the operation to generate the PDF data.

Discussion
The new `NSPrintOperation` object uses the default `NSPrintInfo` object. This method raises an `NSPrintOperationExistsException` if there is already a print operation in progress; otherwise the returned object is made the current print operation for this thread.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ PDFOperationWithView:insideRect:toData:printInfo:` (page 2084)
`+ PDFOperationWithView:insideRect:toPath:printInfo:` (page 2085)
`- runOperation` (page 2093)

Declared In
`NSPrintOperation.h`

PDFOperationWithView:insideRect:toData:printInfo:

Creates and returns a new `NSPrintOperation` object ready to control the copying of PDF graphics from the specified view using the specified print settings.

```
+ (NSPrintOperation *)PDFOperationWithView:(NSView *)aView insideRect:(NSRect)rect
     toData:(NSMutableData *)data printInfo:(NSPrintInfo *)aPrintInfo
```

Parameters

aView

The view containing the data to be turned into PDF data.

rect

The portion of the view (specified in points in the view's coordinate space) to be rendered as PDF data.

data

An empty `NSMutableData` object. After the job is run, this object contains the PDF data.

aPrintInfo

The print settings to use when generating the PDF data.

Return Value
The new `NSPrintOperation` object. You must run the operation to generate the PDF data.

Discussion
This method raises an `NSPrintOperationExistsException` if there is already a print operation in progress; otherwise the returned object is made the current print operation for this thread.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ PDFOperationWithView:insideRect:toData:` (page 2083)

+ PDFOperationWithView:insideRect:toPath:printInfo: (page 2085)
- runOperation (page 2093)

Declared In
NSPrintOperation.h

PDFOperationWithView:insideRect:toPath:printInfo:

Creates and returns a new NSPrintOperation object ready to control the copying of PDF graphics from the specified view and write the resulting data to the specified file.

```
+ (NSPrintOperation *)PDFOperationWithView:(NSView *)aView insideRect:(NSRect)rect
     toPath:(NSString *)path printInfo:(NSPrintInfo *)aPrintInfo
```

Parameters

aView
The view containing the data to be turned into PDF data.

rect
The portion of the view (specified in points in the view's coordinate space) to be rendered as PDF data.

path
The path to a file. After the job is run, this file contains the PDF data.

aPrintInfo
The print settings to use when generating the PDF data.

Return Value
The new NSPrintOperation object. You must run the operation to generate the PDF data.

Discussion
This method raises an NSPrintOperationExistsException if there is already a print operation in progress; otherwise the returned object is made the current print operation for this thread.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ PDFOperationWithView:insideRect:toData: (page 2083)
+ PDFOperationWithView:insideRect:toData:printInfo: (page 2084)
- runOperation (page 2093)

Declared In
NSPrintOperation.h

printOperationWithView:

Creates and returns an NSPrintOperation object ready to control the printing of the specified view.

```
+ (NSPrintOperation *)printOperationWithView:(NSView *)aView
```

Parameters

aView

The view whose contents you want to print.

Return Value

The new `NSPrintOperation` object. You must run the operation to print the view.

Discussion

The new `NSPrintOperation` object uses the settings stored in the shared `NSPrintInfo` object. This method raises an `NSPrintOperationExistsException` if there is already a print operation in progress; otherwise the returned object is made the current print operation for this thread.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ printOperationWithView:printInfo:` (page 2086)

`- runOperation` (page 2093)

Related Sample Code

FunHouse

GLUT

Quartz2DBasics

SimpleToolbar

ToolbarSample

Declared In

`NSPrintOperation.h`

printOperationWithView:printInfo:

Creates and returns an `NSPrintOperation` object ready to control the printing of the specified view using custom print settings.

```
+ (NSPrintOperation *)printOperationWithView:(NSView *)aView printInfo:(NSPrintInfo
    *)aPrintInfo
```

Parameters

aView

The view whose contents you want to print.

aPrintInfo

The print settings to use when printing the view.

Return Value

The new `NSPrintOperation` object. You must run the operation to print the view.

Discussion

This method raises an `NSPrintOperationExistsException` if there is already a print operation in progress; otherwise the returned object is made the current print operation for this thread.

Availability

Available in Mac OS X v10.0 and later.

See Also
`+ printOperationWithView:` (page 2085)
`- runOperation` (page 2093)

Related Sample Code
From A View to A Movie
ImageApp
QTKitPlayer
Quartz Composer WWDC 2005 TextEdit
Sketch-112

Declared In
`NSPrintOperation.h`

setCurrentOperation:

Sets the current print operation for this thread.

```
+ (void)setCurrentOperation:(NSPrintOperation *)operation
```

Parameters
operation
The print operation to make current. You may specify `nil` to clear the current print operation.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ currentOperation` (page 2081)

Declared In
`NSPrintOperation.h`

Instance Methods

accessoryView

Returns the accessory view used by the receiver's print panel. (Deprecated in Mac OS X v10.5. Use the `accessoryControllers` method of `NSPrintPanel` instead.)

```
- (NSView *)accessoryView
```

Return Value
The custom accessory view.

Discussion
You use the `setAccessoryView:` (page 2095) method to customize the default `NSPrintPanel` object without having to subclass `NSPrintPanel` or specify your own print panel object.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

See Also
- `printPanel` (page 2093)
- `setPrintPanel:` (page 2097)
- `setShowPanels:` (page 2098)
- `showPanels` (page 2100)

Declared In
`NSPrintOperation.h`

canSpawnSeparateThread

Returns a Boolean value indicating whether the receiver is allowed to spawn a separate printing thread.

```
- (BOOL)canSpawnSeparateThread
```

Return Value
`YES` if the receiver is allowed to spawn a separate thread; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setCanSpawnSeparateThread:` (page 2095)

Declared In
`NSPrintOperation.h`

cleanUpOperation

Called at the end of a print operation to remove the receiver as the current operation.

```
- (void)cleanUpOperation
```

Discussion
You typically do not invoke this method directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `runOperation` (page 2093)

Declared In
`NSPrintOperation.h`

context

Returns the graphics context object used for generating output.

```
- (NSGraphicsContext *)context
```

Return Value
The graphics context object used for drawing during the operation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `createContext` (page 2089)
- `destroyContext` (page 2090)

Declared In
`NSPrintOperation.h`

createContext

Creates the graphics context object used for drawing during the operation.

```
- (NSGraphicsContext *)createContext
```

Return Value
The graphics context object used for drawing. This object is created using the settings from the receiver's `NSPrintInfo` object.

Discussion
Do not invoke this method directly—it is invoked before any output is generated.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `context` (page 2089)
- `destroyContext` (page 2090)

Declared In
`NSPrintOperation.h`

currentPage

Returns the current page number being printed.

```
- (NSInteger)currentPage
```

Return Value
The current page being printed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `pageOrder` (page 2092)
- `setPageOrder:` (page 2097)

Declared In
`NSPrintOperation.h`

deliverResult

Delivers the results of the print operation to the intended destination.

```
- (BOOL)deliverResult
```

Return Value
`YES` if the results were successfully delivered; otherwise, `NO`.

Discussion
This method may be called to deliver the results to the printer spool or preview application. Do not invoke this method directly—it is invoked automatically when the print operation is done.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `runOperation` (page 2093)

Declared In
`NSPrintOperation.h`

destroyContext

Destroys the receiver's graphics context.

```
- (void)destroyContext
```

Discussion
Do not invoke this method directly—it is invoked at the end of a print operation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `context` (page 2089)
- `createContext` (page 2089)

Declared In
`NSPrintOperation.h`

isCopyingOperation

Returns a Boolean value indicating whether the receiver is an EPS or PDF copy operation.

```
- (BOOL)isCopyingOperation
```

Return Value
`YES` if the receiver is an EPS or PDF copy operation; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPrintOperation.h`

jobStyleHint

Returns the type of content that the print job is printing. (Deprecated in Mac OS X v10.5. Use the `jobStyleHint` method of `NSPrintPanel` instead.)

```
- (NSString *)jobStyleHint
```

Return Value
The content description, or `nil` if no job style hint has been set.

Availability
Available in Mac OS X v10.2 and later.
Deprecated in Mac OS X v10.5.

See Also
`- setJobStyleHint:` (page 2096)

Declared In
`NSPrintOperation.h`

jobTitle

Returns the title of the print job.

```
- (NSString *)jobTitle
```

Return Value
A string containing the print job title. If set, this value overrides the title returned by the printing view.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setJobTitle:` (page 2096)
`- printJobTitle` (page 3198) (NSView)

Declared In
`NSPrintOperation.h`

pageOrder

Returns the print order for the pages.

```
- (NSPrintingPageOrder)pageOrder
```

Return Value
The print order. For a list of possible values, see "Constants" (page 2101).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `currentPage` (page 2089)

Declared In
`NSPrintOperation.h`

pageRange

Returns the range of pages associated with the print operation.

```
- (NSRange)pageRange
```

Return Value
The range of page numbers. Page numbers are one-based values where the index of page one is 1, the index of page two is 2, and so on. Depending on the information returned by the printing view, the starting page number may not be 1. Also, if the number of pages being printed is not known, the page count may be set to `NSIntegerMax`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `knowsPageRange:` (page 3185) (NSView)

Declared In
`NSPrintOperation.h`

printInfo

Returns the receiver's `NSPrintInfo` object.

```
- (NSPrintInfo *)printInfo
```

Return Value
The print settings of the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPrintInfo:` (page 2097)

Related Sample Code
Quartz2DBasics
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSPrintOperation.h`

printPanel

Returns the `NSPrintPanel` object used when running the operation.

```
- (NSPrintPanel *)printPanel
```

Return Value
The print panel object for the operation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `accessoryView` (page 2087)
- `setAccessoryView:` (page 2095)
- `setPrintPanel:` (page 2097)
- `setShowPanels:` (page 2098)
- `showPanels` (page 2100)

Declared In
`NSPrintOperation.h`

runOperation

Runs the print operation on the current thread.

```
- (BOOL)runOperation
```

Return Value
`YES` if the operation was successful; otherwise, `NO`.

Discussion
The operation runs to completion in the current thread, blocking the application. A separate thread is not spawned, even if `canSpawnSeparateThread` (page 2088) is `YES`. Use `runOperationModalForWindow:delegate:didRunSelector:contextInfo:` (page 2094) to use document-modal sheets and to allow a separate thread to perform the operation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cleanUpOperation` (page 2088)
- `deliverResult` (page 2090)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
Quartz2DBasics

Declared In
NSPrintOperation.h

runOperationModalForWindow:delegate:didRunSelector:contextInfo:

Runs the print operation, calling your custom delegate method upon completion.

```
- (void)runOperationModalForWindow:(NSWindow *)docWindow delegate:(id)delegate
    didRunSelector:(SEL)didRunSelector contextInfo:(void *)contextInfo
```

Parameters

docWindow
The document window to receive a print progress sheet.

delegate
The printing delegate object. Messages are sent to this object.

didRunSelector
The delegate method called after the completion of the print operation.

contextInfo
A pointer to any data you want passed to the method in the *didRunSelector* parameter.

Discussion
The method specified by the *didRunSelector* parameter must have the following signature:

```
- (void)printOperationDidRun:(NSPrintOperation *)printOperation
success:(BOOL)success  contextInfo:(void *)contextInfo
```

The value of *success* is YES if the print operation ran to completion without cancellation or error, and NO otherwise.

If you send setCanSpawnSeparateThread: (page 2095) to an NSPrintOperation object with an argument of YES, then the delegate specified in a subsequent invocation of runOperationModalForWindow:delegate:didRunSelector:contextInfo: (page 2094) may be messaged in that spawned, non-main thread.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
GLSL Showpiece Lite
GLUT
Quartz Composer WWDC 2005 TextEdit
SimpleToolbar
ToolbarSample

Declared In
NSPrintOperation.h

setAccessoryView:

Sets the custom accessory view to be displayed by the receiver's print panel. (Deprecated in Mac OS X v10.5. Use the `addAccessoryController:` method of `NSPrintPanel` instead.)

```
- (void)setAccessoryView:(NSView *)aView
```

Parameters

aView

The view to display in the print panel. You can use this view to specify additional print options.

Discussion

You can use this method to avoid subclassing `NSPrintPanel` or specifying your own print panel object. The print panel is automatically resized (as needed) to accommodate the accessory view when it is selected.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

See Also

- `accessoryView` (page 2087)
- `printPanel` (page 2093)
- `setPrintPanel:` (page 2097)
- `setShowPanels:` (page 2098)
- `showPanels` (page 2100)

Declared In

`NSPrintOperation.h`

setCanSpawnSeparateThread:

Sets whether the receiver is allowed to spawn a separate printing thread.

```
- (void)setCanSpawnSeparateThread:(BOOL)canSpawnSeparateThread
```

Parameters

canSpawnSeparateThread

`YES` if the receiver is allowed to spawn a separate thread; otherwise, `NO`.

Discussion

If *canSpawnSeparateThread* is `YES`, an `NSThread` object is detached when the print panel is dismissed (or immediately, if the panel is not to be displayed). The new thread performs the print operation, so that control can return to your application. A thread is detached only if the print operation is run using the `runOperationModalForWindow:delegate:didRunSelector:contextInfo:` (page 2094) method. If *canSpawnSeparateThread* is `NO`, the operation runs on the current thread, blocking the application until the operation completes.

If you send `setCanSpawnSeparateThread:` (page 2095) to an `NSPrintOperation` object with an argument of `YES`, then the delegate specified in a subsequent invocation of `runOperationModalForWindow:delegate:didRunSelector:contextInfo:` (page 2094) may be messaged in that spawned, non-main thread.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `canSpawnSeparateThread` (page 2088)

Related Sample Code
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSPrintOperation.h`

setJobStyleHint:

Sets the type of content that the print job is printing. (Deprecated in Mac OS X v10.5. Use the `setJobStyleHint:` method of `NSPrintPanel` instead.)

```
- (void)setJobStyleHint:(NSString *)hint
```

Parameters

hint

A supported job style hint. Valid values for this parameter are described in the "Constants" (page 2114) section of the `NSPrintPanel` class. If this value is `nil`, the standard interface is used.

Discussion
This controls the set of items that appear in the Presets menu of the simplified Print panel interface presented by this operation, if it presents one.

Availability
Available in Mac OS X v10.2 and later.
Deprecated in Mac OS X v10.5.

See Also
- `jobStyleHint` (page 2091)

Declared In
`NSPrintOperation.h`

setJobTitle:

Assigns a custom title to the print job.

```
- (void)setJobTitle:(NSString *)jobTitle
```

Parameters

jobTitle

The print job title. The receiver makes its own copy of the specified string.

Discussion
Assigning a title with this method overrides the job title provided by the printing view's `printJobTitle` (page 3198) method. Specifying `nil` for the *jobTitle* parameter causes the receiver to once again take its title from the printing view.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `jobTitle` (page 2091)
- `printJobTitle` (page 3198) (NSView)

Declared In
`NSPrintOperation.h`

setPageOrder:

Sets the print order for the pages of the operation.

```
- (void)setPageOrder:(NSPrintingPageOrder)order
```

Parameters

order
The print order. For a list of possible values, see "Constants" (page 2101).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `currentPage` (page 2089)
- `pageOrder` (page 2092)

Declared In
`NSPrintOperation.h`

setPrintInfo:

Sets the receiver's `NSPrintInfo` object.

```
- (void)setPrintInfo:(NSPrintInfo *)aPrintInfo
```

Parameters

aPrintInfo
The new print settings for the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `printInfo` (page 2092)

Declared In
`NSPrintOperation.h`

setPrintPanel:

Sets the `NSPrintPanel` object to be used during the operation.

```
- (void)setPrintPanel:(NSPrintPanel *)panel
```

Parameters

panel

The print panel object to use for the operation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- accessoryView (page 2087)
- printPanel (page 2093)
- setAccessoryView: (page 2095)
- setShowPanels: (page 2098)
- showPanels (page 2100)

Declared In
NSPrintOperation.h

setShowPanels:

Sets whether the print operation should display a print panel. (Deprecated in Mac OS X v10.4 and later. Use setShowsPrintPanel: (page 2099) and setShowsProgressPanel: (page 2099) instead.)

```
- (void)setShowPanels:(BOOL)flag
```

Parameters

flag

YES if the print operation should display a print panel; otherwise, NO.

Discussion
This method also affects whether a progress panel is presented while the operation runs. If an EPS or PDF copy operation is being performed, neither panel is displayed, regardless of the value of *flag*.

Availability
Deprecated in Mac OS X v10.4 and later.

See Also
- accessoryView (page 2087)
- printPanel (page 2093)
- setAccessoryView: (page 2095)
- setPrintPanel: (page 2097)
- showPanels (page 2100)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer
Quartz Composer WWDC 2005 TextEdit

Declared In
NSPrintOperation.h

setShowsPrintPanel:

Sets whether the receiver displays a print panel for this operation.

```
- (void)setShowsPrintPanel:(BOOL)flag
```

Parameters

flag

YES if you want to display a print panel; otherwise, NO.

Discussion
This method does not affect the display of a progress panel; that operation is controlled by the setShowsProgressPanel: (page 2099) method.

Operations that generate EPS or PDF data do no display a progress panel, regardless of the value in the *flag* parameter.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setShowsProgressPanel: (page 2099)
- showsPrintPanel (page 2100)

Declared In
NSPrintOperation.h

setShowsProgressPanel:

Sets whether the receiver displays a progress panel for this operation.

```
- (void)setShowsProgressPanel:(BOOL)flag
```

Parameters

flag

YES if you want to display a progress panel; otherwise, NO.

Discussion
This method does not affect the display of a print panel; that operation is controlled by the setShowsPrintPanel: (page 2099) method.

Operations that generate EPS or PDF data do no display a progress panel, regardless of the value in the *flag* parameter.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setShowsPrintPanel: (page 2099)
- showsProgressPanel (page 2100)

Declared In
NSPrintOperation.h

showPanels

Returns a Boolean value that indicates whether the print panel is to be displayed. (Deprecated in Mac OS X v10.4 and later. Use `showsPrintPanel` (page 2100) and `showsProgressPanel` (page 2100) instead.)

```
- (BOOL)showPanels
```

Return Value
`YES` if the print panel is to be displayed; otherwise, `NO`.

Availability
Deprecated in Mac OS X v10.4 and later.

See Also
- `accessoryView` (page 2087)
- `printPanel` (page 2093)
- `setAccessoryView:` (page 2095)
- `setPrintPanel:` (page 2097)
- `setShowPanels:` (page 2098)

Declared In
NSPrintOperation.h

showsPrintPanel

Returns a Boolean value indicating whether a print panel is displayed during the operation,

```
- (BOOL)showsPrintPanel
```

Return Value
`YES` if the operation displays a print panel; otherwise, `NO`.

Discussion
Operations that generate EPS or PDF data do no display a print panel (instance of `NSPrintPanel`), regardless of the value returned by this method.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setShowsPrintPanel:` (page 2099)

Declared In
NSPrintOperation.h

showsProgressPanel

Returns a Boolean value indicating whether a progress panel is displayed during the operation.

```
- (BOOL)showsProgressPanel
```

Return Value
`YES` if the operation displays a progress panel; otherwise, `NO`.

Discussion
Operations that generate EPS or PDF data do no display a progress panel, regardless of the value returned by this method.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setShowsProgressPanel:` (page 2099)

Declared In
`NSPrintOperation.h`

view

Returns the view object that generates the actual data for the print operation.

```
- (NSView *)view
```

Return Value
The view object that generates the data.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPrintOperation.h`

Constants

NSPrintingPageOrder

These constants specify the page order.

```
typedef enum _NSPrintingPageOrder {
    NSDescendingPageOrder = (-1),
    NSSpecialPageOrder    = 0,
    NSAscendingPageOrder  = 1,
    NSUnknownPageOrder    = 2
} NSPrintingPageOrder;
```

Constants
`NSAscendingPageOrder`
Ascending (back to front) page order.

Available in Mac OS X v10.0 and later.

Declared in `NSPrintOperation.h`.

`NSDescendingPageOrder`
Descending (front to back) page order.
Available in Mac OS X v10.0 and later.
Declared in `NSPrintOperation.h`.

`NSSpecialPageOrder`
The spooler does not rearrange pages—they are printed in the order received by the spooler.
Available in Mac OS X v10.0 and later.
Declared in `NSPrintOperation.h`.

`NSUnknownPageOrder`
No page order specified.
Available in Mac OS X v10.0 and later.
Declared in `NSPrintOperation.h`.

Discussion
These constants are used by `pageOrder` (page 2092) and `setPageOrder:` (page 2097).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPrintOperation.h`

Exception Name

This is the name of an exception that can be raised by `NSPrintOperation`.

```
NSString *NSPrintOperationExistsException;
```

Constants
`NSPrintOperationExistsException`
The name of an exception raised when there is already a print operation in process.
The methods that raise this exception are the `EPSOperation...` and `printOperation...`.
Available in Mac OS X v10.0 and later.
Declared in `NSPrintOperation.h`.

Declared In
`NSPrintOperation.h`

NSPrintPanel Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSPrintPanel.h
Companion guides	Printing Programming Topics for Cocoa Sheet Programming Topics

Overview

An `NSPrintPanel` object creates the Print panel used to query the user for information about a print job. This panel may lets the user select the range of pages to print and the number of copies before executing the Print command.

Print panels can display a simplified interface when printing certain types of data. For example, the panel can display a list of print-setting presets, which lets the user enable print settings in groups as opposed to individually. The `setJobStyleHint:` (page 2113) method activates the simplified interface and identifies which presets to display.

Tasks

Creating an NSPrintPanel

`+ printPanel` (page 2105)
Returns a new `NSPrintPanel` object.

Customizing the Panel

`- options` (page 2109)
Returns the current configuration options for the Print panel.

`- setOptions:` (page 2113)
Sets the configuration options for the Print panel.

- `defaultButtonTitle` (page 2107)
  Returns the title of the Print panel's default button.
- `setDefaultButtonTitle:` (page 2112)
  Sets the title of the Print panel's default button.
- `helpAnchor` (page 2108)
  Returns the HTML help anchor associated with the Print panel.
- `setHelpAnchor:` (page 2113)
  Sets the HTML help anchor for the print panel.
- `jobStyleHint` (page 2109) Deprecated in Mac OS X v10.5
  Returns the type of content that the Print panel is representing.
- `setJobStyleHint:` (page 2113) Deprecated in Mac OS X v10.5
  Sets the type of content the Print panel is representing.

Managing Accessory Views

- `addAccessoryController:` (page 2106)
  Adds a custom controller to the Print panel to manage an accessory view.
- `removeAccessoryController:` (page 2110)
  Removes the specified controller and accessory view from the Print panel.
- `accessoryControllers` (page 2105)
  Returns the array of controller objects used to manage the Print panel's accessory views.

Running the Panel

- `beginSheetWithPrintInfo:modalForWindow:delegate:didEndSelector:contextInfo:` (page 2107)
  Displays a Print panel sheet and runs it modally for the specified window.
- `runModal` (page 2111)
  Displays the receiver's Print panel and begins the modal loop.
- `runModalWithPrintInfo:` (page 2111)
  Displays the receiver's Print panel and runs the modal loop using the specified printing information.

Communicating with the NSPrintInfo Object

- `printInfo` (page 2110)
  Returns the printing information associated with the running Print panel.
- `finalWritePrintInfo` (page 2108) Deprecated in Mac OS X v10.5
  Writes the receiver's printing attributes to the current `NSPrintOperation` object.
- `updateFromPrintInfo` (page 2114) Deprecated in Mac OS X v10.5
  Updates the receiver with information from the current `NSPrintOperation` object.

Deprecated Methods

- `accessoryView` (page 2106)
  Returns the accessory view of the receiver. (Deprecated. Use `accessoryControllers` (page 2105) instead.)
- `setAccessoryView:` (page 2112) Deprecated in Mac OS X v10.5
  Sets the accessory view for the receiver. (Deprecated. Use `addAccessoryController:` (page 2106) instead.)
- `pickedAllPages:` (page 2109) Available in Mac OS X v10.0 through Mac OS X v10.4
  Deprecated. (Deprecated. No alternative. Do not use.)
- `pickedButton:` (page 2109) Available in Mac OS X v10.0 through Mac OS X v10.4
  Deprecated. (Deprecated. No alternative. Do not use.)
- `pickedLayoutList:` (page 2110) Available in Mac OS X v10.0 through Mac OS X v10.4
  Deprecated. (Deprecated. No alternative. Do not use.)

Class Methods

printPanel

Returns a new `NSPrintPanel` object.

```
+ (NSPrintPanel *)printPanel
```

Return Value
The print panel object.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPrintPanel.h`

Instance Methods

accessoryControllers

Returns the array of controller objects used to manage the Print panel's accessory views.

```
- (NSArray *)accessoryControllers
```

Return Value
An array of `NSViewController` objects, each of which manages an accessory view for the Print panel.

Discussion
This method returns the accessory views that were added using the `addAccessoryController:` method.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `addAccessoryController:` (page 2106)

Declared In
`NSPrintPanel.h`

accessoryView

Returns the accessory view of the receiver. (Deprecated in Mac OS X v10.5. Use `accessoryControllers` (page 2105) instead.)

```
- (NSView *)accessoryView
```

Return Value
The accessory view of the receiver, if any.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

See Also
- `setAccessoryView:` (page 2112)

Declared In
`NSPrintPanel.h`

addAccessoryController:

Adds a custom controller to the Print panel to manage an accessory view.

```
- (void)addAccessoryController:(NSViewController < NSPrintPanelAccessorizing >
    *)accessoryController
```

Parameters

accessoryController
The view controller that manages your custom accessory views.

Discussion
You can invoke this method multiple times to add multiple accessory views to the receiver's Print panel.

The title for the accessory view is obtained from the `title` method of the view controller object.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `removeAccessoryController:` (page 2110)

Declared In
`NSPrintPanel.h`

beginSheetWithPrintInfo:modalForWindow:delegate:didEndSelector:contextInfo:

Displays a Print panel sheet and runs it modally for the specified window.

```
- (void)beginSheetWithPrintInfo:(NSPrintInfo *)printInfo modalForWindow:(NSWindow
    *)docWindow delegate:(id)modalDelegate didEndSelector:(SEL)didEndSelector
   contextInfo:(void *)contextInfo
```

Parameters

printInfo
The printing information for the current job.

docWindow
The window on which to display the sheet.

modalDelegate
A modal delegate object assigned to handle the closing of the Print panel sheet.

didEndSelector
The selector to call on the modal delegate object when the sheet is dismissed. The signature of this method is listed in the Discussion section.

contextInfo
A pointer to context data the `didEndSelector` method needs to process the sheet. This data is user-defined and may be `NULL`.

Discussion
When the modal session ends, if *modalDelegate* and *didEndSelector* contain non-`nil` values, the method specified by *didEndSelector* is invoked on the object in *modalDelegate*. The data you specify in *contextInfo* is passed as a parameter to the *didEndSelector* method. The object in *modalDelegate* is not the same as a delegate assigned to the panel. Modal delegates for sheets are temporary and the relationship lasts only until the sheet is dismissed.

The *didEndSelector* argument must have the following signature:

```
- (void)printPanelDidEnd:(NSPrintPanel *)printPanel
returnCode:(NSInteger)returnCode  contextInfo: (void *)contextInfo;
```

The value passed as *returnCode* is either `NSCancelButton` or `NSOKButton`. The value `NSOKButton` is returned even if the user clicked the Preview button.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPrintPanel.h`

defaultButtonTitle

Returns the title of the Print panel's default button.

```
- (NSString *)defaultButtonTitle
```

Return Value
The title of the default button.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setDefaultButtonTitle:` (page 2112)

Declared In
`NSPrintPanel.h`

finalWritePrintInfo

Writes the receiver's printing attributes to the current `NSPrintOperation` object. (Deprecated in Mac OS X v10.5.)

```
- (void)finalWritePrintInfo
```

Discussion
Do not invoke this method directly—it is invoked automatically when the Print panel is dismissed by the user clicking the OK button.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

See Also
- `updateFromPrintInfo` (page 2114)
+ `currentOperation` (page 2081) (NSPrintOperation)

Declared In
`NSPrintPanel.h`

helpAnchor

Returns the HTML help anchor associated with the Print panel.

```
- (NSString *)helpAnchor
```

Return Value
The current help anchor.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setHelpAnchor:` (page 2113)

Declared In
`NSPrintPanel.h`

jobStyleHint

Returns the type of content that the Print panel is representing.

```
- (NSString *)jobStyleHint
```

Return Value
A string containing the job style hint or `nil` if no hint has been set.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- setJobStyleHint:` (page 2113)

Declared In
`NSPrintPanel.h`

options

Returns the current configuration options for the Print panel.

```
- (NSPrintPanelOptions)options
```

Return Value
One or more configuration constants added together. To determine if a particular option is set, AND the return value with the appropriate constants found in "NSPrintPanelOptions" (page 2115).

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setOptions:` (page 2113)

Declared In
`NSPrintPanel.h`

pickedAllPages:

Deprecated. (Available in Mac OS X v10.0 through Mac OS X v10.4. No alternative. Do not use.)

```
- (void)pickedAllPages:(id)sender
```

Availability
Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared In
`NSPrintPanel.h`

pickedButton:

Deprecated. (Available in Mac OS X v10.0 through Mac OS X v10.4. No alternative. Do not use.)

```
- (void)pickedButton:(id)sender
```

Availability
Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared In
`NSPrintPanel.h`

pickedLayoutList:

Deprecated. (Available in Mac OS X v10.0 through Mac OS X v10.4. No alternative. Do not use.)

```
- (void)pickedLayoutList:(id)sender
```

Availability
Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared In
`NSPrintPanel.h`

printInfo

Returns the printing information associated with the running Print panel.

```
- (NSPrintInfo *)printInfo
```

Return Value
The current printing information. May return `nil` if the Print panel is not currently running.

Discussion
This method is a convenience method that your delegate can use to get the printing information while the Print Panel is visible.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPrintPanel.h`

removeAccessoryController:

Removes the specified controller and accessory view from the Print panel.

```
- (void)removeAccessoryController:(NSViewController < NSPrintPanelAccessorizing >
    *)accessoryController
```

Parameters
accessoryController
The view controller to remove.

Discussion
You use this method to remove any view controllers responsible for displaying accessory views you do not want to include in the Print panel.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `addAccessoryController:` (page 2106)

Declared In
`NSPrintPanel.h`

runModal

Displays the receiver's Print panel and begins the modal loop.

```
- (NSInteger)runModal
```

Return Value
`NSCancelButton` if the user clicks the Cancel button; otherwise `NSOKButton`.

Discussion
This method uses the printing information associated with the current printing operation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `printInfo` (page 2092) (NSPrintOperation)

Declared In
`NSPrintPanel.h`

runModalWithPrintInfo:

Displays the receiver's Print panel and runs the modal loop using the specified printing information.

```
- (NSInteger)runModalWithPrintInfo:(NSPrintInfo *)printInfo
```

Parameters
printInfo
The printing information to use while displaying the Print panel.

Return Value
`NSCancelButton` if the user clicks the Cancel button; otherwise `NSOKButton`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPrintPanel.h`

setAccessoryView:

Sets the accessory view for the receiver. (Deprecated in Mac OS X v10.5. Use `addAccessoryController:` (page 2106) instead.)

```
- (void)setAccessoryView:(NSView *)aView
```

Parameters

aView

The view containing the controls and information you want to add to the print panel. Specify `nil` to remove the receiver's current accessory view.

Discussion

You can use an accessory view to add printing controls and information to the standard print panel. You set your accessory view prior to displaying the print panel. Upon display, the print panel adds your application's name as an item to its pane-selection pull-down menu. When the user selects this item, the print panel displays your accessory view.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

See Also

`- accessoryView` (page 2106)

Declared In

`NSPrintPanel.h`

setDefaultButtonTitle:

Sets the title of the Print panel's default button.

```
- (void)setDefaultButtonTitle:(NSString *)defaultButtonTitle
```

Parameters

defaultButtonTitle

The string to use for the button title.

Discussion

You can use this method to change the default button title from "Print" to something more appropriate for your usage of the panel. For example, if you are using the Print panel to save a representation of the document to a file, you might change the title to "Save".

Availability

Available in Mac OS X v10.5 and later.

See Also

`- defaultButtonTitle` (page 2107)

Declared In

`NSPrintPanel.h`

setHelpAnchor:

Sets the HTML help anchor for the print panel.

```
- (void)setHelpAnchor:(NSString *)helpAnchor
```

Parameters

helpAnchor

The anchor name in your Apple Help file. This parameter should contain just the name portion of the HTML anchor element.

Discussion

For information on how to insert anchors into your Apple Help files, see Authoring User Help in *Apple Help Programming Guide*.

Availability

Available in Mac OS X v10.5 and later.

See Also

- helpAnchor (page 2108)

Declared In

NSPrintPanel.h

setJobStyleHint:

Sets the type of content the Print panel is representing.

```
- (void)setJobStyleHint:(NSString *)hint
```

Parameters

hint

For a list of supported job style hints, see "Job Style Hints" (page 2114). Pass nil to this method to deactivate the simplified Print panel interface and use the standard interface instead (the equivalent of Core Printing's kPMPresetGraphicsTypeGeneral).

Discussion

This method controls the set of items that appear in the Presets menu of the simplified Print panel interface.

Availability

Available in Mac OS X v10.2 and later.

See Also

- jobStyleHint (page 2109)

Declared In

NSPrintPanel.h

setOptions:

Sets the configuration options for the Print panel.

```
- (void)setOptions:(NSPrintPanelOptions)options
```

Parameters

options

The configuration options, which you specify by adding together the appropriate constant values found in "NSPrintPanelOptions" (page 2115).

Availability
Available in Mac OS X v10.5 and later.

See Also
- options (page 2109)

Declared In
NSPrintPanel.h

updateFromPrintInfo

Updates the receiver with information from the current NSPrintOperation object. (Deprecated in Mac OS X v10.5.)

```
- (void)updateFromPrintInfo
```

Discussion
Do not invoke this method directly—it is invoked automatically before the Print panel is displayed.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

See Also
- finalWritePrintInfo (page 2108)
+ currentOperation (page 2081) (NSPrintOperation)

Declared In
NSPrintPanel.h

Constants

Job Style Hints

This constant can be passed to the setJobStyleHint: (page 2113) method to activate the simplified Print panel interface and specify which presets to display.

```
NSString *const NSPrintPhotoJobStyleHint;
NSString *const NSPrintAllPresetsJobStyleHint;
NSString *const NSPrintNoPresetsJobStyleHint;
```

Constants

`NSPrintPhotoJobStyleHint`

Output contains photographic data.

Available in Mac OS X v10.2 and later.

Declared in `NSPrintPanel.h`.

`NSPrintAllPresetsJobStyleHint`

Output appropriate to all graphics types. Equivalent to Core Printing's `kPMPresetGraphicsTypeAll`.

Available in Mac OS X v10.6 and later.

Declared in `NSPrintPanel.h`.

`NSPrintNoPresetsJobStyleHint`

Output excludes all graphics printing. Equivalent to Core Printing's `kPMPresetGraphicsTypeNone`.

Available in Mac OS X v10.6 and later.

Declared in `NSPrintPanel.h`.

Declared In

`NSPrintPanel.h`

NSPrintPanelOptions

These constants are used to configure the contents of the main Print panel.

```
enum {
    NSPrintPanelShowsCopies = 0x01,
    NSPrintPanelShowsPageRange = 0x02,
    NSPrintPanelShowsPaperSize = 0x04,
    NSPrintPanelShowsOrientation = 0x08,
    NSPrintPanelShowsScaling = 0x10,
    NSPrintPanelShowsPrintSelection = 1 << 5,
    NSPrintPanelShowsPageSetupAccessory = 0x100,
    NSPrintPanelShowsPreview = 0x20000
};
typedef NSInteger NSPrintPanelOptions;
```

Constants

`NSPrintPanelShowsCopies`

The Print panel includes a field for manipulating the number of copies being printed. This field is separate from any accessory views.

Available in Mac OS X v10.5 and later.

Declared in `NSPrintPanel.h`.

`NSPrintPanelShowsPageRange`

The Print panel includes a set of fields for manipulating the range of pages being printed. These fields are separate from any accessory views.

Available in Mac OS X v10.5 and later.

Declared in `NSPrintPanel.h`.

`NSPrintPanelShowsPaperSize`
: The Print panel includes a control for manipulating the paper size of the printer. This control is separate from any accessory views.

  Available in Mac OS X v10.5 and later.

  Declared in `NSPrintPanel.h`.

`NSPrintPanelShowsOrientation`
: The Print panel includes a control for manipulating the page orientation. This control is separate from any accessory views.

  Available in Mac OS X v10.5 and later.

  Declared in `NSPrintPanel.h`.

`NSPrintPanelShowsScaling`
: The Print panel includes a control for scaling the printed output. This control is separate from any accessory views.

  Available in Mac OS X v10.5 and later.

  Declared in `NSPrintPanel.h`.

`NSPrintPanelShowsPrintSelection`
: The Print panel includes an additional selection option for paper range. This control is separate from any accessory views.

  Available in Mac OS X v10.6 and later.

  Declared in `NSPrintPanel.h`.

`NSPrintPanelShowsPageSetupAccessory`
: The Print panel includes a separate accessory view for manipulating the paper size, orientation, and scaling attributes. Page setup fields that are already configured for display on the main portion of the Print panel appear there and not on this accessory panel.

  Available in Mac OS X v10.5 and later.

  Declared in `NSPrintPanel.h`.

`NSPrintPanelShowsPreview`
: The Print panel displays a built-in preview of the document contents. This option is only appropriate when the Print panel is used in conjunction with an `NSPrintOperation` object to print a document.

  Available in Mac OS X v10.5 and later.

  Declared in `NSPrintPanel.h`.

Declared In
`NSPrintPanel.h`

NSProgressIndicator Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSProgressIndicator.h
Companion guide	Progress Indicators
Related sample code	BackgroundExporter CIVideoDemoGL MyPhoto SimpleStickies URL CacheInfo

Overview

The `NSProgressIndicator` class lets an application display a progress indicator to show that a lengthy task is under way. Some progress indicators are indeterminate and do nothing more than spin to show that the application is busy. Others are determinate and show the percentage of the task that has been completed.

Tasks

Animating the Progress Indicator

- `setUsesThreadedAnimation:` (page 2128)
  Sets a hint as to whether the receiver should implement animation of the progress indicator in a separate thread.
- `startAnimation:` (page 2129)
  Starts the animation of an indeterminate progress indicator.
- `stopAnimation:` (page 2129)
  Stops the animation of an indeterminate progress indicator.

- `usesThreadedAnimation` (page 2130)
  Returns whether the receiver implements the animation of the progress indicator in a separate thread.
- `animate:` (page 2119) Deprecated in Mac OS X v10.5 and later
  This action method advances the progress animation of an indeterminate progress animator by one step. (Deprecated. `NSProgressIndicator` no longer supports incrementing the animation using this method. Use the `startAnimation:` (page 2129) and `stopAnimation:` (page 2129) methods to perform animation of the progress indicator.)
- `animationDelay` (page 2120) Deprecated in Mac OS X v10.5 and later
  Returns the delay, in seconds, between animation steps for an indeterminate progress indicator. (Deprecated. Progress indicators no longer allow the animation delay to be set.)
- `setAnimationDelay:` (page 2123) Deprecated in Mac OS X v10.5 and later
  Sets the delay, in seconds, between animation steps for an indeterminate progress indicator. (Deprecated. Progress indicators no longer allow the animation delay to be set.)

Advancing the Progress Bar

- `incrementBy:` (page 2121)
  Advances the progress bar of a determinate progress indicator by the specified amount.
- `setDoubleValue:` (page 2125)
  Sets the value that indicates the current extent of the receiver.
- `doubleValue` (page 2121)
  Returns a value that indicates the current extent of the progress bar of a determinate progress indicator.
- `setMinValue:` (page 2127)
  Sets the minimum value for the receiver.
- `minValue` (page 2123)
  Returns the minimum value for the progress bar of a determinate progress indicator.
- `setMaxValue:` (page 2126)
  Sets the maximum value for the receiver.
- `maxValue` (page 2123)
  Returns the maximum value for the progress bar of a determinate progress indicator.

Setting the Appearance

- `setControlSize:` (page 2124)
  Sets the size of the receiver.
- `controlSize` (page 2120)
  Returns the size of the receiver.
- `setControlTint:` (page 2124)
  Sets the receiver's control tint.
- `controlTint` (page 2120)
  Returns the receiver's control tint.
- `setBezeled:` (page 2124)
  Sets whether the receiver's frame has a three-dimensional bezel.

- `isBezeled` (page 2121)
  Returns a Boolean value indicating whether the receiver's frame has a bezel.
- `setIndeterminate:` (page 2126)
  Sets whether the receiver is indeterminate.
- `isIndeterminate` (page 2122)
  Returns a Boolean value indicating whether the receiver is indeterminate.
- `setStyle:` (page 2127)
  Sets the style of the progress indicator (bar or spinning).
- `style` (page 2130)
  Returns the style of the progress indicator (bar or spinning).
- `sizeToFit` (page 2128)
  This action method resizes the receiver to an appropriate size depending on what `style` (page 2130) returns.
- `setDisplayedWhenStopped:` (page 2125)
  Sets whether the receiver hides itself when it isn't animating.
- `isDisplayedWhenStopped` (page 2122)
  Returns a Boolean value indicating whether the receiver shows itself even when it's not animating.

Instance Methods

animate:

This action method advances the progress animation of an indeterminate progress animator by one step. (Deprecated in Mac OS X v10.5 and later. `NSProgressIndicator` no longer supports incrementing the animation using this method. Use the `startAnimation:` (page 2129) and `stopAnimation:` (page 2129) methods to perform animation of the progress indicator.)

```
- (void)animate:(id)sender
```

Parameters

sender
The object sending the message.

Availability

Deprecated in Mac OS X v10.5 and later.

See Also

- `startAnimation:` (page 2129)
- `stopAnimation:` (page 2129)

Related Sample Code

MenuItemView

Declared In

`NSProgressIndicator.h`

animationDelay

Returns the delay, in seconds, between animation steps for an indeterminate progress indicator. (Deprecated in Mac OS X v10.5 and later. Progress indicators no longer allow the animation delay to be set.)

```
- (NSTimeInterval)animationDelay
```

Return Value
The delay between animation steps. By default, the animation delay is set to 1/12 of a second (5.0/60.0). A determinate progress indicator does not use the animation delay value.

Availability
Deprecated in Mac OS X v10.5 and later.

Declared In
NSProgressIndicator.h

controlSize

Returns the size of the receiver.

```
- (NSControlSize)controlSize
```

Return Value
A constant indicating the size of the control. Valid return values are described in NSCell (page 533).

Availability
Available in Mac OS X v10.0 and later.

See Also
- setControlSize: (page 2124)

Declared In
NSProgressIndicator.h

controlTint

Returns the receiver's control tint.

```
- (NSControlTint)controlTint
```

Return Value
A constant indicating the current control tint. Valid return values are described in NSCell (page 533).

Availability
Available in Mac OS X v10.0 and later.

See Also
- setControlTint: (page 2124)

Declared In
NSProgressIndicator.h

doubleValue

Returns a value that indicates the current extent of the progress bar of a determinate progress indicator.

```
- (double)doubleValue
```

Return Value
The value representing the current extent of a determinate progress bar. For example, a determinate progress indicator goes from 0.0 to 100.0 by default. If the progress bar has advanced halfway across the view, the value returned by `doubleValue` would be 50.0. An indeterminate progress indicator does not use this value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `incrementBy:` (page 2121)
- `setDoubleValue:` (page 2125)

Declared In
`NSProgressIndicator.h`

incrementBy:

Advances the progress bar of a determinate progress indicator by the specified amount.

```
- (void)incrementBy:(double)delta
```

Parameters
delta
The amount by which to increment the progress bar. For example, if you want to advance a progress bar from 0.0 to 100.0 in 20 steps, you would invoke `incrementBy:` (page 2121) 20 times with a delta value of 5.0.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `doubleValue` (page 2121)

Declared In
`NSProgressIndicator.h`

isBezeled

Returns a Boolean value indicating whether the receiver's frame has a bezel.

```
- (BOOL)isBezeled
```

Return Value
`YES` if the receiver's frame has a three-dimensional bezel; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setBezeled: (page 2124)

Declared In
NSProgressIndicator.h

isDisplayedWhenStopped

Returns a Boolean value indicating whether the receiver shows itself even when it's not animating.

```
- (BOOL)isDisplayedWhenStopped
```

Return Value
YES if the progress indicator shows itself even when it's not animating. By default, this returns returns YES if style (page 2130) is NSProgressIndicatorBarStyle and NO if style (page 2130) isNSProgressIndicatorSpinningStyle.

Availability
Available in Mac OS X v10.2 and later.

See Also
- setDisplayedWhenStopped: (page 2125)

Declared In
NSProgressIndicator.h

isIndeterminate

Returns a Boolean value indicating whether the receiver is indeterminate.

```
- (BOOL)isIndeterminate
```

Return Value
YES if the progress bar is indeterminate; otherwise NO. This applies only if style (page 2130) returns NSProgressIndicatorBarStyle.

Discussion
If style (page 2130) returns NSProgressIndicatorSpinningStyle, the indicator is always indeterminate, regardless of what this method returns.

A determinate indicator displays how much of the task has been completed. An indeterminate indicator shows simply that the application is busy.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setIndeterminate: (page 2126)

Declared In
NSProgressIndicator.h

maxValue

Returns the maximum value for the progress bar of a determinate progress indicator.

```
- (double)maxValue
```

Return Value
The maximum value of the progress indicator. By default, a determinate progress indicator goes from 0.0 to 100.0, so the value returned would be 100.0. An indeterminate progress indicator does not use this value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `minValue` (page 2123)
- `setMaxValue:` (page 2126)

Declared In
`NSProgressIndicator.h`

minValue

Returns the minimum value for the progress bar of a determinate progress indicator.

```
- (double)minValue
```

Return Value
The minimum value of the progress indicator. By default, a determinate progress indicator goes from 0.0 to 100.0, so the value returned would be 0.0. An indeterminate progress indicator does not use this value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `maxValue` (page 2123)
- `setMinValue:` (page 2127)

Declared In
`NSProgressIndicator.h`

setAnimationDelay:

Sets the delay, in seconds, between animation steps for an indeterminate progress indicator. (Deprecated in Mac OS X v10.5 and later. Progress indicators no longer allow the animation delay to be set.)

```
- (void)setAnimationDelay:(NSTimeInterval)delay
```

Parameters

delay

The number of seconds between animation steps. By default, the animation delay is set to 1/12 of a second (5.0/60.0). Setting the delay to a `double` value larger than 5.0/60.0 slows the animation, while setting the delay to a smaller value speeds it up. A determinate progress indicator does not use the animation delay value.

Availability
Deprecated in Mac OS X v10.5 and later.

Declared In
NSProgressIndicator.h

setBezeled:

Sets whether the receiver's frame has a three-dimensional bezel.

```
- (void)setBezeled:(BOOL)flag
```

Parameters

flag

YES if the progress indicator is bezeled; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isBezeled (page 2121)

Declared In
NSProgressIndicator.h

setControlSize:

Sets the size of the receiver.

```
- (void)setControlSize:(NSControlSize)size
```

Parameters

size

A constant indicating the size of the control. Valid values for *size* are described in NSCell (page 533).

Availability
Available in Mac OS X v10.0 and later.

See Also
- controlSize (page 2120)

Related Sample Code
AnimatedTableView

Declared In
NSProgressIndicator.h

setControlTint:

Sets the receiver's control tint.

```
- (void)setControlTint:(NSControlTint)controlTint
```

Parameters

controlTint

A constant indicating the desired control tint. Valid values for *controlTint* are described in NSCell (page 533).

Availability
Available in Mac OS X v10.0 and later.

See Also
- controlTint (page 2120)

Declared In
NSProgressIndicator.h

setDisplayedWhenStopped:

Sets whether the receiver hides itself when it isn't animating.

```
- (void)setDisplayedWhenStopped:(BOOL)isDisplayed
```

Parameters

isDisplayed

YES to hide the progress indicator when it isn't animating; otherwise NO. By default, this is YES if style (page 2130) is NSProgressIndicatorBarStyle, and NO if style (page 2130) is NSProgressIndicatorSpinningStyle.

Availability
Available in Mac OS X v10.2 and later.

See Also
- isDisplayedWhenStopped (page 2122)

Related Sample Code
ScannerBrowser

Declared In
NSProgressIndicator.h

setDoubleValue:

Sets the value that indicates the current extent of the receiver.

```
- (void)setDoubleValue:(double)doubleValue
```

Parameters

doubleValue

The current extent of a determinate progress indicator.

Discussion
An indeterminate progress indicator does not use this value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `doubleValue` (page 2121)
- `incrementBy:` (page 2121)
- `setMaxValue:` (page 2126)
- `setMinValue:` (page 2127)

Related Sample Code
CIVideoDemoGL
EnhancedAudioBurn
ExtractMovieAudioToAIFF
From A View to A Movie
QTExtractAndConvertToAIFF

Declared In
`NSProgressIndicator.h`

setIndeterminate:

Sets whether the receiver is indeterminate.

```
- (void)setIndeterminate:(BOOL)flag
```

Parameters

flag

`YES` if the progress indicator should be indeterminate; otherwise `NO`.

Discussion
This method only has an effect if `style` (page 2130) returns `NSProgressIndicatorBarStyle`. If `style` (page 2130) returns `NSProgressIndicatorSpinningStyle`, the indicator is always indeterminate, regardless of what you pass to this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isIndeterminate` (page 2122)

Related Sample Code
AnimatedTableView
ExtractMovieAudioToAIFF
From A View to A Movie
QTExtractAndConvertToAIFF
ScannerBrowser

Declared In
`NSProgressIndicator.h`

setMaxValue:

Sets the maximum value for the receiver.

```
- (void)setMaxValue:(double)newMaximum
```

Parameters

newMaximum

The maximum value of the progress indicator. An indeterminate progress indicator does not use this value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- maxValue (page 2123)

Related Sample Code
EnhancedAudioBurn
From A View to A Movie

Declared In
NSProgressIndicator.h

setMinValue:

Sets the minimum value for the receiver.

```
- (void)setMinValue:(double)newMinimum
```

Parameters

newMinimum

The minimum value of the progress indicator. An indeterminate progress indicator does not use this value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- minValue (page 2123)

Related Sample Code
EnhancedAudioBurn
From A View to A Movie

Declared In
NSProgressIndicator.h

setStyle:

Sets the style of the progress indicator (bar or spinning).

```
- (void)setStyle:(NSProgressIndicatorStyle)style
```

Parameters

style

A constant indicating the style of the progress indicator. Possible values for *style* are described in `NSProgressIndicatorStyle` (page 2131).

Availability
Available in Mac OS X v10.2 and later.

See Also
`- style` (page 2130)

Related Sample Code
AnimatedTableView
From A View to A Movie
SourceView

Declared In
`NSProgressIndicator.h`

setUsesThreadedAnimation:

Sets a hint as to whether the receiver should implement animation of the progress indicator in a separate thread.

```
- (void)setUsesThreadedAnimation:(BOOL)flag
```

Parameters

flag

`YES` to indicate that animation of the progress indicator should occur in a separate thread; otherwise, `NO`. This value is only a hint and may be ignored.

Discussion
If the application becomes multithreaded as a result of an invocation of this method, the application's performance could become noticeably slower.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- usesThreadedAnimation` (page 2130)

Related Sample Code
SimpleStickies

Declared In
`NSProgressIndicator.h`

sizeToFit

This action method resizes the receiver to an appropriate size depending on what `style` (page 2130) returns.

```
- (void)sizeToFit
```

Discussion
Use this after you use `setStyle:` (page 2127) to re-size the receiver.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
AnimatedTableView
From A View to A Movie

Declared In
`NSProgressIndicator.h`

startAnimation:

Starts the animation of an indeterminate progress indicator.

```
- (void)startAnimation:(id)sender
```

Parameters
sender
The object sending the message.

Discussion
Does nothing for a determinate progress indicator.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `animationDelay` (page 2120)
- `stopAnimation:` (page 2129)

Related Sample Code
AnimatedTableView
ExtractMovieAudioToAIFF
SampleScannerApp
ScriptingBridgeFinder
SimpleStickies

Declared In
`NSProgressIndicator.h`

stopAnimation:

Stops the animation of an indeterminate progress indicator.

```
- (void)stopAnimation:(id)sender
```

Parameters
sender
The object sending the message.

Discussion
Does nothing for a determinate progress indicator.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `animationDelay` (page 2120)
- `startAnimation:` (page 2129)

Related Sample Code
CAPlayThrough
ExtractMovieAudioToAIFF
QTExtractAndConvertToAIFF
SampleScannerApp
SimpleStickies

Declared In
`NSProgressIndicator.h`

style

Returns the style of the progress indicator (bar or spinning).

```
- (NSProgressIndicatorStyle)style
```

Return Value
A constant indicating the style of the progress indicator. Possible return values are described in NSProgressIndicatorStyle (page 2131).

Availability
Available in Mac OS X v10.2 and later.

See Also
- `setStyle:` (page 2127)

Declared In
`NSProgressIndicator.h`

usesThreadedAnimation

Returns whether the receiver implements the animation of the progress indicator in a separate thread.

```
- (BOOL)usesThreadedAnimation
```

Return Value
`YES` if animation of the progress indicator occurs in a separate thread; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setUsesThreadedAnimation:` (page 2128)

Declared In
`NSProgressIndicator.h`

Constants

NSProgressIndicatorThickness

Specify the height of a progress indicator.

```
typedef enum _NSProgressIndicatorThickness {
    NSProgressIndicatorPreferredThickness       = 14,
    NSProgressIndicatorPreferredSmallThickness  = 10,
    NSProgressIndicatorPreferredLargeThickness  = 18,
    NSProgressIndicatorPreferredAquaThickness   = 12
} NSProgressIndicatorThickness;
```

Constants

`NSProgressIndicatorPreferredThickness`
14
Available in Mac OS X v10.0 and later.
Declared in `NSProgressIndicator.h`.

`NSProgressIndicatorPreferredSmallThickness`
10
Available in Mac OS X v10.0 and later.
Declared in `NSProgressIndicator.h`.

`NSProgressIndicatorPreferredLargeThickness`
18
Available in Mac OS X v10.0 and later.
Declared in `NSProgressIndicator.h`.

`NSProgressIndicatorPreferredAquaThickness`
12
Available in Mac OS X v10.0 and later.
Declared in `NSProgressIndicator.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSProgressIndicator.h`

NSProgressIndicatorStyle

Specify the progress indicator's style and are used by `style` (page 2130) and `setStyle:` (page 2127).

```
typedef enum _NSProgressIndicatorStyle {
    NSProgressIndicatorBarStyle = 0,
    NSProgressIndicatorSpinningStyle = 1
} NSProgressIndicatorStyle;
```

Constants

`NSProgressIndicatorBarStyle`

A rectangular indicator that can be determinate or indeterminate.

Available in Mac OS X v10.2 and later.

Declared in `NSProgressIndicator.h`.

`NSProgressIndicatorSpinningStyle`

A small square indicator that can be indeterminate only .

Available in Mac OS X v10.2 and later.

Declared in `NSProgressIndicator.h`.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSProgressIndicator.h`

NSResponder Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSResponder.h AppKit/NSInterfaceStyle.h
Companion guide	Cocoa Event-Handling Guide
Related sample code	AnimatedTableView LayerBackedOpenGLView LightTable NSOpenGL Fullscreen SourceView

Overview

`NSResponder` is an abstract class that forms the basis of event and command processing in the Application Kit. The core classes—`NSApplication`, `NSWindow`, and `NSView`—inherit from `NSResponder`, as must any class that handles events. The responder model is built around three components: event messages, action messages, and the responder chain.

Starting with Mac OS X v10.4, `NSResponder` plays an important role in the presentation of error information. The default implementations of the `presentError:` (page 2187) and `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 2187) methods send `willPresentError:` (page 2205) to `self`, thereby giving subclasses the opportunity to customize the localized information presented in error alerts. `NSResponder` then forwards the message to the next responder, passing it the customized `NSError` object. The exact path up the modified responder chain depends on the type of application window:

- Windows owned by document: view to superviews to window to window controller to document object to document controller to the application object
- Windows with window controllers but no documents: view to superviews to window to window controller to the application object
- Windows with no window controllers: view to superviews to window to the application object

`NSApplication` displays a document-modal error alert and, if the error object has a recovery attempter, gives it a chance to recover from the error. (A recovery attempter is an object that conforms to the `NSErrorRecoveryAttempting` informal protocol.)

Tasks

Changing the First Responder

- `acceptsFirstResponder` (page 2143)
  Overridden by subclasses to return `YES` if the receiver accepts first responder status.
- `becomeFirstResponder` (page 2144)
  Notifies the receiver that it's about to become first responder in its `NSWindow`.
- `resignFirstResponder` (page 2189)
  Notifies the receiver that it's been asked to relinquish its status as first responder in its window.

Managing the Next Responder

- `setNextResponder:` (page 2197)
  Sets the receiver's next responder.
- `nextResponder` (page 2182)
  Returns the receiver's next responder, or `nil` if it has none.

Responding to Mouse Events

- `mouseDown:` (page 2164)
  Informs the receiver that the user has pressed the left mouse button.
- `mouseDragged:` (page 2164)
  Informs the receiver that the user has moved the mouse with the left button pressed.
- `mouseUp:` (page 2166)
  Informs the receiver that the user has released the left mouse button.
- `mouseMoved:` (page 2166)
  Informs the receiver that the mouse has moved.
- `mouseEntered:` (page 2165)
  Informs the receiver that the cursor has entered a tracking rectangle.
- `mouseExited:` (page 2165)
  Informs the receiver that the cursor has exited a tracking rectangle.
- `rightMouseDown:` (page 2189)
  Informs the receiver that the user has pressed the right mouse button.
- `rightMouseDragged:` (page 2190)
  Informs the receiver that the user has moved the mouse with the right button pressed .
- `rightMouseUp:` (page 2190)
  Informs the receiver that the user has released the right mouse button.

- `otherMouseDown:` (page 2183)
  Informs the receiver that the user has pressed a mouse button other than the left or right one.
- `otherMouseDragged:` (page 2183)
  Informs the receiver that the user has moved the mouse with a button other than the left or right button pressed.
- `otherMouseUp:` (page 2184)
  Informs the receiver that the user has released a mouse button other than the left or right button.

Responding to Key Events

- `keyDown:` (page 2159)
  Informs the receiver that the user has pressed a key.
- `keyUp:` (page 2159)
  Informs the receiver that the user has released a key.
- `interpretKeyEvents:` (page 2158)
  Invoked by subclasses from their `keyDown:` (page 2159) method to handle a series of key events.
- `performKeyEquivalent:` (page 2186)
  Overridden by subclasses to handle a key equivalent.
- `performMnemonic:` (page 2186)
  Overridden by subclasses to handle a mnemonic.
- `flushBufferedKeyEvents` (page 2153)
  Overridden by subclasses to clear any unprocessed key events.

Responding to Other Kinds of Events

- `cursorUpdate:` (page 2147)
  Informs the receiver that the mouse cursor has moved into a cursor rectangle.
- `flagsChanged:` (page 2152)
  Informs the receiver that the user has pressed or released a modifier key (Shift, Control, and so on).
- `tabletPoint:` (page 2199)
  Informs the receiver that a tablet-point event has occurred.
- `tabletProximity:` (page 2200)
  Informs the receiver that a tablet-proximity event has occurred.
- `helpRequested:` (page 2153)
  Displays context-sensitive help for the receiver if such exists; otherwise passes this message to the next responder.
- `scrollWheel:` (page 2193)
  Informs the receiver that the mouse's scroll wheel has moved.

Responding to Action Messages

- `cancelOperation:` (page 2145)
  Implemented by subclasses to cancel the current operation.

- `capitalizeWord:` (page 2145)
  Implemented by subclasses to capitalize the word or words surrounding the insertion point or selection, expanding the selection if necessary.
- `centerSelectionInVisibleArea:` (page 2146)
  Implemented by subclasses to scroll the selection, whatever it is, inside its visible area.
- `changeCaseOfLetter:` (page 2146)
  Implemented by subclasses to change the case of a letter or letters in the selection, perhaps by opening a panel with capitalization options or by cycling through possible case combinations.
- `complete:` (page 2147)
  Implemented by subclasses to complete an operation in progress or a partially constructed element.
- `deleteBackward:` (page 2148)
  Implemented by subclasses to delete the selection, if there is one, or a single element backward from the insertion point (a letter or character in text, for example).
- `deleteBackwardByDecomposingPreviousCharacter:` (page 2148)
  Implemented by subclasses to delete the selection, if there is one, or a single character backward from the insertion point.
- `deleteForward:` (page 2148)
  Implemented by subclasses to delete the selection, if there is one, or a single element forward from the insertion point (a letter or character in text, for example).
- `deleteToBeginningOfLine:` (page 2149)
  Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the beginning of a line (typically of text).
- `deleteToBeginningOfParagraph:` (page 2149)
  Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the beginning of a paragraph of text.
- `deleteToEndOfLine:` (page 2150)
  Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the end of a line (typically of text).
- `deleteToEndOfParagraph:` (page 2150)
  Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the end of a paragraph of text.
- `deleteToMark:` (page 2150)
  Implemented by subclasses to delete the selection, if there is one, or all items from the insertion point to a previously placed mark, including the selection itself if not empty.
- `deleteWordBackward:` (page 2151)
  Implemented by subclasses to delete the selection, if there is one, or a single word backward from the insertion point.
- `deleteWordForward:` (page 2151)
  Implemented by subclasses to delete the selection, if there is one, or a single word forward from the insertion point.
- `indent:` (page 2153)
  Implemented by subclasses to indent the selection or the insertion point if there is no selection.
- `insertBacktab:` (page 2154)
  Implemented by subclasses to handle a backward tab.

- `insertContainerBreak:` (page 2154)
  Implemented by subclasses to insert a container break (typically a page break) at the insertion point or selection, deleting the selection if there is one.
- `insertLineBreak:` (page 2155)
  Implemented by subclasses to insert a line break (as distinguished from a paragraph break) at the insertion point or selection, deleting the selection if there is one.
- `insertNewline:` (page 2155)
  Implemented by subclasses to insert a newline character at the insertion point or selection, deleting the selection if there is one, or to end editing if the receiver is a text field or other field editor.
- `insertNewlineIgnoringFieldEditor:` (page 2156)
  Implemented by subclasses to insert a line-break character at the insertion point or selection, deleting the selection if there is one.
- `insertParagraphSeparator:` (page 2156)
  Implemented by subclasses to insert a paragraph separator at the insertion point or selection, deleting the selection if there is one.
- `insertTab:` (page 2157)
  Implemented by subclasses to insert a tab character at the insertion point or selection, deleting the selection if there is one, or to end editing if the receiver is a text field or other field editor.
- `insertTabIgnoringFieldEditor:` (page 2157)
  Implemented by subclasses to insert a tab character at the insertion point or selection, deleting the selection if there is one.
- `insertDoubleQuoteIgnoringSubstitution:` (page 2155)
  Implemented by subclasses to insert a double quote character at the insertion point without interference by automatic quote correction.
- `insertSingleQuoteIgnoringSubstitution:` (page 2156)
  Implemented by subclasses to insert a single quote character at the insertion point without interference by automatic quote correction.
- `insertText:` (page 2158)
  Overridden by subclasses to insert the supplied string at the insertion point or selection, deleting the selection if there is one.
- `lowercaseWord:` (page 2160)
  Implemented by subclasses to make lowercase every letter in the word or words surrounding the insertion point or selection, expanding the selection if necessary.
- `moveBackward:` (page 2166)
  Implemented by subclasses to move the selection or insertion point one element or character backward.
- `moveBackwardAndModifySelection:` (page 2167)
  Implemented by subclasses to expand or reduce either end of the selection backward by one element or character.
- `moveParagraphForwardAndModifySelection:` (page 2170)
  Implemented by subclasses to move the selection or insertion point to the beginning of the next paragraph, expanding or reducing the current selection.
- `moveParagraphBackwardAndModifySelection:` (page 2170)
  Implemented by subclasses to move the selection or insertion point to the beginning of the previous paragraph, expanding or reducing the current selection.
- `moveToBeginningOfDocumentAndModifySelection:` (page 2172)
  Implemented by subclasses to move the selection or insertion point to the beginning of the document, expanding or reducing the current selection.

- `moveToEndOfDocumentAndModifySelection:` (page 2174)
  Implemented by subclasses to move the selection or insertion point to the end of the document, expanding or reducing the current selection.
- `moveToBeginningOfLineAndModifySelection:` (page 2173)
  Implemented by subclasses to move the selection or insertion point to the beginning of the line, expanding or reducing the current selection.
- `moveToEndOfLineAndModifySelection:` (page 2175)
  Implemented by subclasses to move the selection or insertion point to the end of the line, expanding or reducing the current selection.
- `moveToBeginningOfParagraphAndModifySelection:` (page 2174)
  Implemented by subclasses to move the selection or insertion point to the beginning of the current paragraph, expanding or reducing the current selection.
- `moveToEndOfParagraphAndModifySelection:` (page 2176)
  Implemented by subclasses to move the selection or insertion point to the end of the current paragraph, expanding or reducing the current selection.
- `moveToLeftEndOfLine:` (page 2176)
  Implemented by subclasses to move the selection or insertion point to the left end of the line.
- `moveToLeftEndOfLineAndModifySelection:` (page 2176)
  Implemented by subclasses to move the selection or insertion point to the left end of the line, expanding or contracting the selection as required.
- `moveToRightEndOfLine:` (page 2177)
  Implemented by subclasses to move the selection or insertion point to the right end of the line
- `moveToRightEndOfLineAndModifySelection:` (page 2177)
  Implemented by subclasses to move the selection or insertion point to the right end of the line, expanding or contracting the selection as required.
- `moveDown:` (page 2167)
  Implemented by subclasses to move the selection or insertion point one element or character down.
- `moveDownAndModifySelection:` (page 2168)
  Implemented by subclasses to expand or reduce the top or bottom end of the selection downward by one element, character, or line (whichever is appropriate for text direction).
- `moveForward:` (page 2168)
  Implemented by subclasses to move the selection or insertion point one element or character forward.
- `moveForwardAndModifySelection:` (page 2168)
  Implemented by subclasses to expand or reduce either end of the selection forward by one element or character.
- `moveLeft:` (page 2169)
  Implemented by subclasses to move the selection or insertion point one element or character to the left.
- `moveLeftAndModifySelection:` (page 2169)
  Implemented by subclasses to expand or reduce either end of the selection to the left (display order) by one element or character.
- `moveRight:` (page 2171)
  Implemented by subclasses to move the selection or insertion point one element or character to the right.

- `moveRightAndModifySelection:` (page 2171)
  Implemented by subclasses to expand or reduce either end of the selection to the right (display order) by one element or character.
- `moveToBeginningOfDocument:` (page 2172)
  Implemented by subclasses to move the selection to the first element of the document or the insertion point to the beginning.
- `moveToBeginningOfLine:` (page 2172)
  Implemented by subclasses to move the selection to the first element of the selected line or the insertion point to the beginning of the line.
- `moveToBeginningOfParagraph:` (page 2173)
  Implemented by subclasses to move the insertion point to the beginning of the selected paragraph.
- `moveToEndOfDocument:` (page 2174)
  Implemented by subclasses to move the selection to the last element of the document or the insertion point to the end.
- `moveToEndOfLine:` (page 2175)
  Implemented by subclasses to move the selection to the last element of the selected line or the insertion point to the end of the line.
- `moveToEndOfParagraph:` (page 2175)
  Implemented by subclasses to move the insertion point to the end of the selected paragraph.
- `moveUp:` (page 2177)
  Implemented by subclasses to move the selection or insertion point one element or character up.
- `moveUpAndModifySelection:` (page 2178)
  Implemented by subclasses to expand or reduce the top or bottom end of the selection upward by one element, character, or line (whichever is appropriate for text direction).
- `moveWordBackward:` (page 2178)
  Implemented by subclasses to move the selection or insertion point one word backward.
- `moveWordBackwardAndModifySelection:` (page 2179)
  Implemented by subclasses to expand or reduce either end of the selection backward by one whole word.
- `moveWordForward:` (page 2179)
  Implemented by subclasses to move the selection or insertion point one word forward, in logical order.
- `moveWordForwardAndModifySelection:` (page 2180)
  Implemented by subclasses to expand or reduce either end of the selection forward by one whole word.
- `moveWordLeft:` (page 2180)
  Implemented by subclasses to move the selection or insertion point one word to the left, in display order.
- `moveWordRight:` (page 2181)
  Implemented by subclasses to move the selection or insertion point one word right.
- `moveWordRightAndModifySelection:` (page 2182)
  Implemented by subclasses to expand or reduce either end of the selection to the right by one whole word.
- `moveWordLeftAndModifySelection:` (page 2181)
  Implemented by subclasses to expand or reduce either end of the selection left by one whole word in display order.

- `pageDown:` (page 2184)
  Implemented by subclasses to scroll the receiver down (or back) one page in its scroll view, also moving the insertion point to the top of the newly displayed page.
- `pageDownAndModifySelection:` (page 2184)
  Implemented by subclasses to scroll the receiver down (or back) one page in its scroll view, also moving the insertion point to the top of the newly displayed page. The selection is expanded or contracted as required.
- `pageUp:` (page 2185)
  Implemented by subclasses to scroll the receiver up (or forward) one page in its scroll view, also moving the insertion point to the top of the newly displayed page.
- `pageUpAndModifySelection:` (page 2185)
  Implemented by subclasses to scroll the receiver up (or forward) one page in its scroll view, also moving the insertion point to the top of the newly displayed page. The selection is expanded or contracted as necessary.
- `scrollToBeginningOfDocument:` (page 2192)
  Implemented by subclasses to scroll the receiver to the beginning of the document, without changing the selection.
- `scrollToEndOfDocument:` (page 2193)
  Implemented by subclasses to scroll the receiver to the end of the document, without changing the selection.
- `scrollLineUp:` (page 2191)
  Implemented by subclasses to scroll the receiver one line up in its scroll view, without changing the selection.
- `scrollLineDown:` (page 2191)
  Implemented by subclasses to scroll the receiver one line down in its scroll view, without changing the selection.
- `scrollPageUp:` (page 2192)
  Implemented by subclasses to scroll the receiver one page up in its scroll view, without changing the selection.
- `scrollPageDown:` (page 2192)
  Implemented by subclasses to scroll the receiver one page down in its scroll view, without changing the selection.
- `selectAll:` (page 2194)
  Implemented by subclasses to select all selectable elements.
- `selectLine:` (page 2194)
  Implemented by subclasses to select all elements in the line or lines containing the selection or insertion point.
- `selectParagraph:` (page 2194)
  Implemented by subclasses to select all paragraphs containing the selection or insertion point.
- `selectToMark:` (page 2195)
  Implemented by subclasses to select all items from the insertion point or selection to a previously placed mark, including the selection itself if not empty.
- `selectWord:` (page 2195)
  Implemented by subclasses to extend the selection to the nearest word boundaries outside it (up to, but not including, word delimiters).

- `setMark:` (page 2196)
  Implemented by subclasses to set a mark at the insertion point or selection, which is used by `deleteToMark:` (page 2150) and `selectToMark:` (page 2195).
- `showContextHelp:` (page 2198)
  Implemented by subclasses to invoke the help system, displaying information relevant to the receiver and its current state.
- `swapWithMark:` (page 2198)
  Swaps the mark and the selection or insertion point, so that what was marked is now the selection or insertion point, and what was the insertion point or selection is now the mark.
- `transpose:` (page 2202)
  Transposes the characters to either side of the insertion point and advances the insertion point past both of them. Does nothing to a selected range of text.
- `transposeWords:` (page 2202)
  Transposes the words to either side of the insertion point and advances the insertion point past both of them. Does nothing to a selected range of text.
- `uppercaseWord:` (page 2203)
  Implemented by subclasses to make uppercase every letter in the word or words surrounding the insertion point or selection, expanding the selection if necessary.
- `yank:` (page 2205)
  Replaces the insertion point or selection with text from the kill buffer.

Presenting and Customizing Error Information

- `presentError:` (page 2187)
  Presents an error alert to the user as an application-modal dialog.
- `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 2187)
  Presents an error alert to the user as a document-modal sheet attached to document window.
- `willPresentError:` (page 2205)
  Implemented by subclasses to return a custom version of the supplied error object that is more suitable for presentation in alert sheets and dialogs.

Dispatching Messages

- `doCommandBySelector:` (page 2152)
  Attempts to perform the indicated command.
- `tryToPerform:with:` (page 2202)
  Attempts to perform the action indicated method with a specified argument.

Managing a Responder's Menu

- `setMenu:` (page 2196)
  Sets the receiver's menu.
- `menu` (page 2163)
  Returns the receiver's menu.

Updating the Services Menu

- `validRequestorForSendType:returnType:` (page 2204)
  Overridden by subclasses to determine what services are available.

Getting the Undo Manager

- `undoManager` (page 2203)
  Returns the undo manager for this responder.

Testing Events

- `shouldBeTreatedAsInkEvent:` (page 2197)
  Returns `YES` if the specified event should be treated as an ink event, `NO` if it should be treated as a mouse event.

Terminating the Responder Chain

- `noResponderFor:` (page 2182)
  Handles the case where an event or action message falls off the end of the responder chain.

Setting the Interface Style

- `setInterfaceStyle:` (page 2195)
  Sets the receiver's style to the style specified by *`interfaceStyle`*, such as `NSMacintoshInterfaceStyle` or `NSWindows95InterfaceStyle`.
- `interfaceStyle` (page 2158)
  Returns the receiver's interface style.

Touch and Gesture Events

- `beginGestureWithEvent:` (page 2144)
  Informs the receiver that the user has begun a touch gesture.
- `endGestureWithEvent:` (page 2152)
  Informs the receiver that the user has ended a touch gesture.
- `magnifyWithEvent:` (page 2160)
  Informs the receiver that the user has begun a pinch gesture.
- `rotateWithEvent:` (page 2190)
  Informs the receiver that the user has begun a rotation gesture.
- `swipeWithEvent:` (page 2199)
  Informs the receiver that the user has begun a swipe gesture.
- `touchesBeganWithEvent:` (page 2200)
  Informs the receiver that new set of touches has been recognized.

- `touchesMovedWithEvent:` (page 2201)
  Informs the receiver that one or more touches has moved.
- `touchesCancelledWithEvent:` (page 2200)
  Informs the receiver that tracking of touches has been cancelled for any reason..
- `touchesEndedWithEvent:` (page 2201)
  Informs the receiver that a set of touches have been removed.

Setting the Writing Direction

- `makeBaseWritingDirectionLeftToRight:` (page 2161)
  Sets the paragraph base writing direction to be left to right.
- `makeBaseWritingDirectionNatural:` (page 2161)
  Sets the paragraph base writing direction to be natural.
- `makeBaseWritingDirectionRightToLeft:` (page 2162)
  Sets the paragraph base writing direction to be right to left.
- `makeTextWritingDirectionLeftToRight:` (page 2162)
  Sets the character level attributed string direction attribute for left to right text.
- `makeTextWritingDirectionNatural:` (page 2163)
  Removes the character-level writing direction attribute
- `makeTextWritingDirectionRightToLeft:` (page 2163)
  Sets the character-level writing direction attribute to a single right-to-left embedding.

Instance Methods

acceptsFirstResponder

Overridden by subclasses to return `YES` if the receiver accepts first responder status.

```
- (BOOL)acceptsFirstResponder
```

Discussion
As first responder, the receiver is the first object in the responder chain to be sent key events and action messages. The `NSResponder` implementation returns `NO`, indicating that by default a responder object doesn't agree to become first responder.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `becomeFirstResponder` (page 2144)
- `resignFirstResponder` (page 2189)

`needsPanelToBecomeKey` (page 3191) (NSView)

Related Sample Code
BindingsJoystick
OpenCL NBody Simulation Example

Sketch-112
SurfaceVertexProgram
VertexPerformanceDemo

Declared In
NSResponder.h

becomeFirstResponder

Notifies the receiver that it's about to become first responder in its NSWindow.

```
- (BOOL)becomeFirstResponder
```

Discussion
The default implementation returns YES, accepting first responder status. Subclasses can override this method to update state or perform some action such as highlighting the selection, or to return NO, refusing first responder status.

Use the NSWindow makeFirstResponder: (page 3344) method, not this method, to make an object the first responder. Never invoke this method directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- resignFirstResponder (page 2189)
- acceptsFirstResponder (page 2143)

Related Sample Code
FilterDemo
NURBSSurfaceVertexProg
Sketch-112
SurfaceVertexProgram
VertexPerformanceDemo

Declared In
NSResponder.h

beginGestureWithEvent:

Informs the receiver that the user has begun a touch gesture.

```
- (void)beginGestureWithEvent:(NSEvent *)event
```

Parameters
event
An event object representing the gesture beginning.

Discussion
The event will be sent to the view under the touch in the key window.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSResponder.h`

cancelOperation:

Implemented by subclasses to cancel the current operation.

```
- (void)cancelOperation:(id)sender
```

Parameters

sender

The object invoking this method.

Discussion
This method is bound to the Escape and Command-. (period) keys. The key window first searches the view hierarchy for a view whose key equivalent is Escape or Command-., whichever was entered. If none of these views handles the key equivalent, the window sends a default action message of `cancelOperation:` to the first responder and from there the message travels up the responder chain.

If no responder in the responder chain implements `cancelOperation:`, the key window searches the view hierarchy for a view whose key equivalent is Escape (note that this may be redundant if the original key equivalent was Escape). If no such responder is found, then a `cancel:` action message is sent to the first responder in the responder chain that implements it.

`NSResponder` declares but does not implement this method.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSResponder.h`

capitalizeWord:

Implemented by subclasses to capitalize the word or words surrounding the insertion point or selection, expanding the selection if necessary.

```
- (void)capitalizeWord:(id)sender
```

Parameters

sender

The object invoking the method.

Discussion
If either end of the selection partially covers a word, that entire word is made lowercase. The *sender* argument is typically the object that invoked this method. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lowercaseWord:` (page 2160)
- `uppercaseWord:` (page 2203)
- `changeCaseOfLetter:` (page 2146)

Declared In
`NSResponder.h`

centerSelectionInVisibleArea:

Implemented by subclasses to scroll the selection, whatever it is, inside its visible area.

```
- (void)centerSelectionInVisibleArea:(id)sender
```

Parameters

sender

The object that invoked the method (typically).

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `scrollLineDown:` (page 2191)
- `scrollLineUp:` (page 2191)
- `scrollPageDown:` (page 2192)
- `scrollPageUp:` (page 2192)

Declared In
`NSResponder.h`

changeCaseOfLetter:

Implemented by subclasses to change the case of a letter or letters in the selection, perhaps by opening a panel with capitalization options or by cycling through possible case combinations.

```
- (void)changeCaseOfLetter:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lowercaseWord:` (page 2160)

- uppercaseWord: (page 2203)
- capitalizeWord: (page 2145)

Declared In
NSResponder.h

complete:

Implemented by subclasses to complete an operation in progress or a partially constructed element.

```
- (void)complete:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
This method can be interpreted, for example, as a request to attempt expansion of a partial word, such as for expanding a glossary shortcut, or to close a graphics item being drawn. NSResponder declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

cursorUpdate:

Informs the receiver that the mouse cursor has moved into a cursor rectangle.

```
- (void)cursorUpdate:(NSEvent *)event
```

Parameters

event

An object encapsulating information about the cursor-update event (NSCursorUpdate (page 1094)).

Discussion
Override this method to set the cursor image. The default implementation uses cursor rectangles, if cursor rectangles are currently valid. If they are not, it calls super to send the message up the responder chain.

If the responder implements this method, but decides not to handle a particular event, it should invoke the superclass implementation of this method.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSResponder.h

deleteBackward:

Implemented by subclasses to delete the selection, if there is one, or a single element backward from the insertion point (a letter or character in text, for example).

```
- (void)deleteBackward:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

deleteBackwardByDecomposingPreviousCharacter:

Implemented by subclasses to delete the selection, if there is one, or a single character backward from the insertion point.

```
- (void)deleteBackwardByDecomposingPreviousCharacter:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

If the previous character is canonically decomposable, this method should try to delete only the last character in the grapheme cluster (for example, deleting "a"+ "´" results in "a"). `NSResponder` declares but does not implement this method.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`NSResponder.h`

deleteForward:

Implemented by subclasses to delete the selection, if there is one, or a single element forward from the insertion point (a letter or character in text, for example).

```
- (void)deleteForward:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

deleteToBeginningOfLine:

Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the beginning of a line (typically of text).

```
- (void)deleteToBeginningOfLine:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
Also places the deleted text into the kill buffer. NSResponder declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- yank: (page 2205)

Declared In
NSResponder.h

deleteToBeginningOfParagraph:

Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the beginning of a paragraph of text.

```
- (void)deleteToBeginningOfParagraph:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
Also places the deleted text into the kill buffer. NSResponder declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- yank: (page 2205)

Declared In
NSResponder.h

deleteToEndOfLine:

Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the end of a line (typically of text).

```
- (void)deleteToEndOfLine:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

Also places the deleted text into the kill buffer. `NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

deleteToEndOfParagraph:

Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the end of a paragraph of text.

```
- (void)deleteToEndOfParagraph:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

Also places the deleted text into the kill buffer. `NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- yank:` (page 2205)

Declared In

`NSResponder.h`

deleteToMark:

Implemented by subclasses to delete the selection, if there is one, or all items from the insertion point to a previously placed mark, including the selection itself if not empty.

```
- (void)deleteToMark:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
Also places the deleted text into the kill buffer. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMark:` (page 2196)
- `selectToMark:` (page 2195)
- `yank:` (page 2205)

Declared In
`NSResponder.h`

deleteWordBackward:

Implemented by subclasses to delete the selection, if there is one, or a single word backward from the insertion point.

```
- (void)deleteWordBackward:(id)sender
```

Parameters
sender
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

deleteWordForward:

Implemented by subclasses to delete the selection, if there is one, or a single word forward from the insertion point.

```
- (void)deleteWordForward:(id)sender
```

Parameters
sender
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

doCommandBySelector:

Attempts to perform the indicated command.

```
- (void)doCommandBySelector:(SEL)aSelector
```

Parameters

aSelector

The selector identifying the method.

Discussion

If the receiver responds to *aSelector*, it invokes the method with `nil` as the argument. If the receiver doesn't respond, it sends this message to its next responder with the same selector. `NSWindow` and `NSApplication` also send the message to their delegates. If the receiver has no next responder or delegate, it beeps.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- tryToPerform:with:` (page 2202)

`sendAction:to:from:` (page 166) (NSApplication)

Declared In

`NSResponder.h`

endGestureWithEvent:

Informs the receiver that the user has ended a touch gesture.

```
- (void)endGestureWithEvent:(NSEvent *)event
```

Parameters

event

An event object representing the gesture end.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

flagsChanged:

Informs the receiver that the user has pressed or released a modifier key (Shift, Control, and so on).

```
- (void)flagsChanged:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the modifier-key event.

Discussion

The default implementation simply passes this message to the next responder.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

flushBufferedKeyEvents

Overridden by subclasses to clear any unprocessed key events.

```
- (void)flushBufferedKeyEvents
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

helpRequested:

Displays context-sensitive help for the receiver if such exists; otherwise passes this message to the next responder.

```
- (void)helpRequested:(NSEvent *)theEvent
```

Parameters

theEvent
An object encapsulating information about the help-request event.

Discussion
NSWindow invokes this method automatically when the user clicks for help—while processing *theEvent*. Subclasses need not override this method, and application code shouldn't directly invoke it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- showContextHelp: (page 2198)

Declared In
NSResponder.h

indent:

Implemented by subclasses to indent the selection or the insertion point if there is no selection.

```
- (void)indent:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

insertBacktab:

Implemented by subclasses to handle a backward tab.

```
- (void)insertBacktab:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
A field editor might respond to this method by selecting the field before it, while a regular text object either doesn't respond to or ignores such a message. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

insertContainerBreak:

Implemented by subclasses to insert a container break (typically a page break) at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertContainerBreak:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method. `NSTextView` implements it to insert an `NSFormFeedCharacter` character (0x000c).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSResponder.h`

insertDoubleQuoteIgnoringSubstitution:

Implemented by subclasses to insert a double quote character at the insertion point without interference by automatic quote correction.

```
- (void)insertDoubleQuoteIgnoringSubstitution:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

insertLineBreak:

Implemented by subclasses to insert a line break (as distinguished from a paragraph break) at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertLineBreak:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method. `NSTextView` implements it to insert an `NSLineSeparatorCharacter` character (0x2028).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSResponder.h`

insertNewline:

Implemented by subclasses to insert a newline character at the insertion point or selection, deleting the selection if there is one, or to end editing if the receiver is a text field or other field editor.

```
- (void)insertNewline:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

insertNewlineIgnoringFieldEditor:

Implemented by subclasses to insert a line-break character at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertNewlineIgnoringFieldEditor:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
Unlike `insertNewline:` (page 2155), this method always inserts a line-break character and doesn't cause the receiver to end editing. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

insertParagraphSeparator:

Implemented by subclasses to insert a paragraph separator at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertParagraphSeparator:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

insertSingleQuoteIgnoringSubstitution:

Implemented by subclasses to insert a single quote character at the insertion point without interference by automatic quote correction.

```
- (void)insertSingleQuoteIgnoringSubstitution:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

insertTab:

Implemented by subclasses to insert a tab character at the insertion point or selection, deleting the selection if there is one, or to end editing if the receiver is a text field or other field editor.

```
- (void)insertTab:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

insertTabIgnoringFieldEditor:

Implemented by subclasses to insert a tab character at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertTabIgnoringFieldEditor:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

Unlike `insertTab:` (page 2157), this method always inserts a tab character and doesn't cause the receiver to end editing. `NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

insertText:

Overridden by subclasses to insert the supplied string at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertText:(id)aString
```

Parameters

aString

The string to insert or replace the selection with. *aString* can be either an `NSString` object or an `NSAttributedString` object.

Discussion

This method is often invoked by the system input manager after the receiver sends a interpretKeyEvents: (page 2158) message. The `NSResponder` implementation simply passes this message to the next responder, or beeps if there is no next responder.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

interfaceStyle

Returns the receiver's interface style.

```
- (NSInterfaceStyle)interfaceStyle
```

Discussion

`interfaceStyle` is an abstract method in `NSResponder` and just returns `NSNoInterfaceStyle`. It is overridden in classes such as `NSWindow` and `NSView` to return the interface style, such as `NSMacintoshInterfaceStyle`. A responder's style (if other than `NSNoInterfaceStyle`) overrides all other settings, such as those established by the defaults system.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setInterfaceStyle:` (page 2195)

Declared In

`NSInterfaceStyle.h`

interpretKeyEvents:

Invoked by subclasses from their `keyDown:` (page 2159) method to handle a series of key events.

```
- (void)interpretKeyEvents:(NSArray *)eventArray
```

Parameters

eventArray

An array of key-event characters to give to the system input manager.

Discussion
This method sends the character input in *eventArray* to the system input manager for interpretation as text to insert or commands to perform. The input manager responds to the request by sending `insertText:` (page 2158) and `doCommandBySelector:` (page 2152) messages back to the invoker of this method. Subclasses shouldn't override this method.

See the `NSInputManager` and `NSTextInput` class and protocol specifications for more information on input management.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

keyDown:

Informs the receiver that the user has pressed a key.

```
- (void)keyDown:(NSEvent *)theEvent
```

Parameters
theEvent
An object encapsulating information about the key-down event.

Discussion
The receiver can interpret *theEvent* itself, or pass it to the system input manager using `interpretKeyEvents:` (page 2158). The default implementation simply passes this message to the next responder.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DragItemAround
GLFullScreen
NSOpenGL Fullscreen
QTNoStepsDemo

Declared In
`NSResponder.h`

keyUp:

Informs the receiver that the user has released a key.

```
- (void)keyUp:(NSEvent *)theEvent
```

Parameters
theEvent
An object encapsulating information about the key-up event.

Discussion
The default implementation simply passes this message to the next responder.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

lowercaseWord:

Implemented by subclasses to make lowercase every letter in the word or words surrounding the insertion point or selection, expanding the selection if necessary.

```
- (void)lowercaseWord:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
If either end of the selection partially covers a word, that entire word is made lowercase. NSResponder declares, but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- uppercaseWord: (page 2203)
- capitalizeWord: (page 2145)
- changeCaseOfLetter: (page 2146)

Declared In
NSResponder.h

magnifyWithEvent:

Informs the receiver that the user has begun a pinch gesture.

```
- (void)magnifyWithEvent:(NSEvent *)event
```

Parameters

event

An event object representing the magnify gesture.

Discussion
The event will be sent to the view under the touch in the key window.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSResponder.h

makeBaseWritingDirectionLeftToRight:

Sets the paragraph base writing direction to be left to right.

```
- (void)makeBaseWritingDirectionLeftToRight:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

Sets the `NSAttributedString` key `NSWritingDirectionAttributeName` to `NSWritingDirectionLeftToRight`.

This action method is intended to be used both as the target of a menu item and for key bindings. The base writing direction methods should be the target of three menu items in a submenu, under the Edit menu.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

makeBaseWritingDirectionNatural:

Sets the paragraph base writing direction to be natural.

```
- (void)makeBaseWritingDirectionNatural:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

Natural directionality is determined from the text in accordance with the Unicode bi-di algorithm. For more information see `NSParagraphStyle`.

Sets the `NSAttributedString`key `NSWritingDirectionAttributeName` to `NSTextWritingDirectionEmbedding`.

This action method is intended to be used both as the target of a menu item and for key bindings. The base writing direction methods should be the target of three menu items in a submenu, under the Edit menu.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

makeBaseWritingDirectionRightToLeft:

Sets the paragraph base writing direction to be right to left.

```
- (void)makeBaseWritingDirectionRightToLeft:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

Sets the `NSAttributedString` key `NSWritingDirectionAttributeName` to `NSWritingDirectionRightToLeft`.

This action method is intended to be used both as the target of a menu item and for key bindings. The base writing direction methods should be the target of three menu items in a submenu, under the Edit menu.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

makeTextWritingDirectionLeftToRight:

Sets the character level attributed string direction attribute for left to right text.

```
- (void)makeTextWritingDirectionLeftToRight:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

Sets the `NSAttributedString` `NSWritingDirectionAttributeName` to `NSWritingDirectionLeftToRight`.

This action method is intended to be used both as the target of a menu item and for key bindings. The text writing directions should be the target of three similar menu items in a submenu under the Edit menu.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

makeTextWritingDirectionNatural:

Removes the character-level writing direction attribute

```
- (void)makeTextWritingDirectionNatural:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

Removes the `NSWritingDirectionAttributeName` from an `NSAttributedString`.

This action method is intended to be used both as the target of a menu item and for key bindings. The text writing directions should be the target of three similar menu items in a submenu under the Edit menu.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

makeTextWritingDirectionRightToLeft:

Sets the character-level writing direction attribute to a single right-to-left embedding.

```
- (void)makeTextWritingDirectionRightToLeft:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

Sets the `NSAttributedString` key `NSWritingDirectionAttributeName` to `NSWritingDirectionRightToLeft`.

This action method is intended to be used both as the target of a menu item and for key bindings. The text writing directions should be the target of three similar menu items in a submenu under the Edit menu.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

menu

Returns the receiver's menu.

```
- (NSMenu *)menu
```

Discussion
For `NSApplication` this menu is the same as the menu returned by its `mainMenu` (page 151) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMenu:` (page 2196)
`menuForEvent:` (page 3189) (NSView)
`defaultMenu` (page 3137) (NSView)

Related Sample Code
ButtonMadness
GLUT
MenuItemView
QTAudioContextInsert
UIElementInspector

Declared In
`NSResponder.h`

mouseDown:

Informs the receiver that the user has pressed the left mouse button.

```
- (void)mouseDown:(NSEvent *)theEvent
```

Parameters
theEvent
An object encapsulating information about the mouse-down event.

Discussion
The default implementation simply passes this message to the next responder.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
GLFullScreen
NSOpenGL Fullscreen
Sketch-112

Declared In
`NSResponder.h`

mouseDragged:

Informs the receiver that the user has moved the mouse with the left button pressed.

```
- (void)mouseDragged:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the mouse-dragged event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

GLFullScreen

NSOpenGL Fullscreen

Declared In

NSResponder.h

mouseEntered:

Informs the receiver that the cursor has entered a tracking rectangle.

```
- (void)mouseEntered:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the mouse-entered event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSResponder.h

mouseExited:

Informs the receiver that the cursor has exited a tracking rectangle.

```
- (void)mouseExited:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the mouse-exited event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSResponder.h

mouseMoved:

Informs the receiver that the mouse has moved.

```
- (void)mouseMoved:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the mouse-moved event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.0 and later.

See Also

`setAcceptsMouseMovedEvents:` (page 3366) (NSWindow)

Declared In

`NSResponder.h`

mouseUp:

Informs the receiver that the user has released the left mouse button.

```
- (void)mouseUp:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the mouse-up event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

GLFullScreen

NSOpenGL Fullscreen

Declared In

`NSResponder.h`

moveBackward:

Implemented by subclasses to move the selection or insertion point one element or character backward.

```
- (void)moveBackward:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
In text, if there is a selection it should be deselected, and the insertion point should be placed at the beginning of the former selection. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

moveBackwardAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection backward by one element or character.

```
- (void)moveBackwardAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
If the end being modified is the backward end, this method expands the selection; if the end being modified is the forward end, it reduces the selection. The first `moveBackwardAndModifySelection:` or `moveForwardAndModifySelection:` (page 2168) method in a series determines the end being modified by always expanding. Hence, this method results in the backward end becoming the mobile one if invoked first. By default, `moveLeftAndModifySelection:` (page 2169) is bound to the left arrow key.

`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

moveDown:

Implemented by subclasses to move the selection or insertion point one element or character down.

```
- (void)moveDown:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
In text, if there is a selection it should be deselected, and the insertion point should be placed below the beginning of the former selection. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

moveDownAndModifySelection:

Implemented by subclasses to expand or reduce the top or bottom end of the selection downward by one element, character, or line (whichever is appropriate for text direction).

```
- (void)moveDownAndModifySelection:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
If the end being modified is the bottom, this method expands the selection; if the end being modified is the top, it reduces the selection. The first `moveDownAndModifySelection:` or `moveUpAndModifySelection:` (page 2178) method in a series determines the end being modified by always expanding. Hence, this method results in the bottom end becoming the mobile one if invoked first.

`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

moveForward:

Implemented by subclasses to move the selection or insertion point one element or character forward.

```
- (void)moveForward:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
In text, if there is a selection it should be deselected, and the insertion point should be placed at the end of the former selection. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

moveForwardAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection forward by one element or character.

```
- (void)moveForwardAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

If the end being modified is the backward end, this method reduces the selection; if the end being modified is the forward end, it expands the selection. The first `moveBackwardAndModifySelection:` (page 2167) or `moveForwardAndModifySelection:` method in a series determines the end being modified by always expanding. Hence, this method results in the forward end becoming the mobile one if invoked first. By default, `moveRightAndModifySelection:` (page 2171) is bound to the right arrow key.

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

moveLeft:

Implemented by subclasses to move the selection or insertion point one element or character to the left.

```
- (void)moveLeft:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

In text, if there is a selection it should be deselected, and the insertion point should be placed at the left end of the former selection. `NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

moveLeftAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection to the left (display order) by one element or character.

```
- (void)moveLeftAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
If the end being modified is the left end, this method expands the selection; if the end being modified is the right end, it reduces the selection. The first `moveLeftAndModifySelection:` or `moveRightAndModifySelection:` (page 2171) method in a series determines the end being modified by always expanding. Hence, this method results in the left end becoming the mobile one if invoked first. By default, this method is bound to the left arrow key.

`NSResponder` declares but doesn't implement this method.

The essential difference between this method and the corresponding `moveBackwardAndModifySelection:` (page 2167) is that the latter method moves in logical order, which can differ in bidirectional text, whereas this method moves in display order.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSResponder.h`

moveParagraphBackwardAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the beginning of the previous paragraph, expanding or reducing the current selection.

```
- (void)moveParagraphBackwardAndModifySelection:(id)sender
```

Parameters
sender
Typically the object that invoked this method.

Discussion
If the cursor is already at the beginning of a paragraph, the selection moves backward to the beginning of the previous paragraph.

`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSResponder.h`

moveParagraphForwardAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the beginning of the next paragraph, expanding or reducing the current selection.

```
- (void)moveParagraphForwardAndModifySelection:(id)sender
```

Parameters
sender
Typically the object that invoked this method.

Discussion
If the cursor is already at the end of a paragraph, the selection moves forward to the end of the next paragraph.

`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSResponder.h`

moveRight:

Implemented by subclasses to move the selection or insertion point one element or character to the right.

```
- (void)moveRight:(id)sender
```

Parameters
sender
Typically the object that invoked this method.

Discussion
In text, if there is a selection it should be deselected, and the insertion point should be placed at the right end of the former selection. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

moveRightAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection to the right (display order) by one element or character.

```
- (void)moveRightAndModifySelection:(id)sender
```

Parameters
sender
Typically the object that invoked this method.

Discussion
If the end being modified is the left end, this method reduces the selection; if the end being modified is the right end, it expands the selection. The first `moveLeftAndModifySelection:` (page 2169) or `moveRightAndModifySelection:` method in a series determines the end being modified by always expanding. Hence, this method results in the right end becoming the mobile one if invoked first. By default, this method is bound to the right arrow key.

`NSResponder` declares but doesn't implement this method.

The essential difference between this method and the corresponding `moveForwardAndModifySelection:` (page 2168) is that the latter method moves in logical order, which can differ in bidirectional text, whereas this method moves in display order.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSResponder.h`

moveToBeginningOfDocument:

Implemented by subclasses to move the selection to the first element of the document or the insertion point to the beginning.

```
- (void)moveToBeginningOfDocument:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

moveToBeginningOfDocumentAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the beginning of the document, expanding or reducing the current selection.

```
- (void)moveToBeginningOfDocumentAndModifySelection:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSResponder.h`

moveToBeginningOfLine:

Implemented by subclasses to move the selection to the first element of the selected line or the insertion point to the beginning of the line.

```
- (void)moveToBeginningOfLine:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

moveToBeginningOfLineAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the beginning of the line, expanding or reducing the current selection.

```
- (void)moveToBeginningOfLineAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

moveToBeginningOfParagraph:

Implemented by subclasses to move the insertion point to the beginning of the selected paragraph.

```
- (void)moveToBeginningOfParagraph:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

moveToBeginningOfParagraphAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the beginning of the current paragraph, expanding or reducing the current selection.

```
- (void)moveToBeginningOfParagraphAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

moveToEndOfDocument:

Implemented by subclasses to move the selection to the last element of the document or the insertion point to the end.

```
- (void)moveToEndOfDocument:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

moveToEndOfDocumentAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the end of the document, expanding or reducing the current selection.

```
- (void)moveToEndOfDocumentAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Availability

Available in Mac OS X v10.6 and later.

Declared In
NSResponder.h

moveToEndOfLine:

Implemented by subclasses to move the selection to the last element of the selected line or the insertion point to the end of the line.

```
- (void)moveToEndOfLine:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
NSResponder declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

moveToEndOfLineAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the end of the line, expanding or reducing the current selection.

```
- (void)moveToEndOfLineAndModifySelection:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
NSResponder declares but doesn't implement this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSResponder.h

moveToEndOfParagraph:

Implemented by subclasses to move the insertion point to the end of the selected paragraph.

```
- (void)moveToEndOfParagraph:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

moveToEndOfParagraphAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the end of the current paragraph, expanding or reducing the current selection.

```
- (void)moveToEndOfParagraphAndModifySelection:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSResponder.h`

moveToLeftEndOfLine:

Implemented by subclasses to move the selection or insertion point to the left end of the line.

```
- (void)moveToLeftEndOfLine:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
In text, if there is a selection it should be deselected, and the insertion point should be placed at left end of the line. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSResponder.h`

moveToLeftEndOfLineAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the left end of the line, expanding or contracting the selection as required.

```
- (void)moveToLeftEndOfLineAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSResponder.h

moveToRightEndOfLine:

Implemented by subclasses to move the selection or insertion point to the right end of the line

```
- (void)moveToRightEndOfLine:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSResponder.h

moveToRightEndOfLineAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the right end of the line, expanding or contracting the selection as required.

```
- (void)moveToRightEndOfLineAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSResponder.h

moveUp:

Implemented by subclasses to move the selection or insertion point one element or character up.

```
- (void)moveUp:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

In text, if there is a selection it should be deselected, and the insertion point should be placed above the beginning of the former selection. `NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

moveUpAndModifySelection:

Implemented by subclasses to expand or reduce the top or bottom end of the selection upward by one element, character, or line (whichever is appropriate for text direction).

```
- (void)moveUpAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

If the end being modified is the bottom, this method reduces the selection; if the end being modified is the top, it expands the selection. The first `moveDownAndModifySelection:` (page 2168) or `moveUpAndModifySelection:` method in a series determines the end being modified by always expanding. Hence, this method results in the top end becoming the mobile one if invoked first.

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

moveWordBackward:

Implemented by subclasses to move the selection or insertion point one word backward.

```
- (void)moveWordBackward:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

If there is a selection it should be deselected, and the insertion point should be placed at the end of the first word preceding the former selection. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `moveWordLeft:` (page 2180)

Declared In
`NSResponder.h`

moveWordBackwardAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection backward by one whole word.

```
- (void)moveWordBackwardAndModifySelection:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
If the end being modified is the backward end, this method expands the selection; if the end being modified is the forward end, it reduces the selection. The first `moveWordBackwardAndModifySelection:` or `moveWordForwardAndModifySelection:` (page 2180) method in a series determines the end being modified by always expanding. Hence, this method results in the backward end becoming the mobile one if invoked first.

`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `moveWordLeftAndModifySelection:` (page 2181)

Declared In
`NSResponder.h`

moveWordForward:

Implemented by subclasses to move the selection or insertion point one word forward, in logical order.

```
- (void)moveWordForward:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
If there is a selection it should be deselected, and the insertion point should be placed at the beginning of the first word following the former selection. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `moveWordRight:` (page 2181)

Declared In
`NSResponder.h`

moveWordForwardAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection forward by one whole word.

```
- (void)moveWordForwardAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
If the end being modified is the backward end, this method reduces the selection; if the end being modified is the forward end, it expands the selection. The first `moveWordBackwardAndModifySelection:` (page 2179) or `moveWordForwardAndModifySelection:` method in a series determines the end being modified by always expanding. Hence, this method results in the forward end becoming the mobile one if invoked first. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `moveWordRightAndModifySelection:` (page 2182)

Declared In
`NSResponder.h`

moveWordLeft:

Implemented by subclasses to move the selection or insertion point one word to the left, in display order.

```
- (void)moveWordLeft:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
If there is a selection it should be deselected, and the insertion point should be placed at the end of the first word to the left of the former selection. `NSResponder` declares but doesn't implement this method.

The main difference between this method and the corresponding `moveWordBackward:` (page 2178) method is that the latter moves in logical order, which is important in bidirectional text, whereas this method moves in display order.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSResponder.h

moveWordLeftAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection left by one whole word in display order.

```
- (void)moveWordLeftAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
If the end being modified is the left end, this method expands the selection; if the end being modified is the right end, it reduces the selection. The first `moveWordLeftAndModifySelection:` or `moveWordRightAndModifySelection:` (page 2182) method in a series determines the end being modified by always expanding. Hence, this method results in the left end becoming the mobile one if invoked first.

`NSResponder` declares but doesn't implement this method.

The main difference between this method and the corresponding `moveWordBackwardAndModifySelection:` (page 2179) method is that the latter moves in logical order, which is important in bidirectional text, whereas this method moves in display order.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSResponder.h

moveWordRight:

Implemented by subclasses to move the selection or insertion point one word right.

```
- (void)moveWordRight:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
If there is a selection it should be deselected, and the insertion point should be placed at the beginning of the first word to the right of the former selection. `NSResponder` declares but doesn't implement this method.

The main difference between this method and the corresponding `moveWordForward:` (page 2179) method is that the latter moves in logical order, which is important in bidirectional text, whereas this method moves in display order.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSResponder.h

moveWordRightAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection to the right by one whole word.

```
- (void)moveWordRightAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
If the end being modified is the backward end, this method reduces the selection; if the end being modified is the forward end, it expands the selection. The first `moveWordBackwardAndModifySelection:` (page 2179) or `moveWordForwardAndModifySelection:` method in a series determines the end being modified by always expanding. Hence, this method results in the forward end becoming the mobile one if invoked first. `NSResponder` declares but doesn't implement this method.

The main difference between this method and the corresponding `moveWordForwardAndModifySelection:` (page 2180) method is that the latter moves in logical order, which is important in bidirectional text, whereas this method moves in display order.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSResponder.h

nextResponder

Returns the receiver's next responder, or `nil` if it has none.

```
- (NSResponder *)nextResponder
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setNextResponder:` (page 2197)
- `noResponderFor:` (page 2182)

Declared In
NSResponder.h

noResponderFor:

Handles the case where an event or action message falls off the end of the responder chain.

```
- (void)noResponderFor:(SEL)eventSelector
```

Parameters

eventSelector

A selector identifying the action or event message.

Discussion

The default implementation beeps if *eventSelector* is `keyDown:` (page 2159).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

otherMouseDown:

Informs the receiver that the user has pressed a mouse button other than the left or right one.

```
- (void)otherMouseDown:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the mouse-down event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`NSResponder.h`

otherMouseDragged:

Informs the receiver that the user has moved the mouse with a button other than the left or right button pressed.

```
- (void)otherMouseDragged:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the mouse-dragged event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`NSResponder.h`

otherMouseUp:

Informs the receiver that the user has released a mouse button other than the left or right button.

```
- (void)otherMouseUp:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the mouse-up event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.1 and later.

Declared In

NSResponder.h

pageDown:

Implemented by subclasses to scroll the receiver down (or back) one page in its scroll view, also moving the insertion point to the top of the newly displayed page.

```
- (void)pageDown:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

NSResponder declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- scrollPageDown: (page 2192)
- scrollPageUp: (page 2192)

Declared In

NSResponder.h

pageDownAndModifySelection:

Implemented by subclasses to scroll the receiver down (or back) one page in its scroll view, also moving the insertion point to the top of the newly displayed page. The selection is expanded or contracted as required.

```
- (void)pageDownAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSResponder.h`

pageUp:

Implemented by subclasses to scroll the receiver up (or forward) one page in its scroll view, also moving the insertion point to the top of the newly displayed page.

```
- (void)pageUp:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `scrollPageDown:` (page 2192)
- `scrollPageUp:` (page 2192)

Declared In
`NSResponder.h`

pageUpAndModifySelection:

Implemented by subclasses to scroll the receiver up (or forward) one page in its scroll view, also moving the insertion point to the top of the newly displayed page. The selection is expanded or contracted as necessary.

```
- (void)pageUpAndModifySelection:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSResponder.h`

performKeyEquivalent:

Overridden by subclasses to handle a key equivalent.

```
- (BOOL)performKeyEquivalent:(NSEvent *)theEvent
```

Parameters

theEvent

An event object that represents the key equivalent pressed.

Discussion

If the character code or codes in *theEvent* match the receiver's key equivalent, the receiver should respond to the event and return `YES`. The default implementation does nothing and returns `NO`.

Note: `performKeyEquivalent:` (page 2186) takes an `NSEvent` object as its argument, while `performMnemonic:` (page 2186) takes an `NSString` object containing the uninterpreted characters of the key event. You should extract the characters for a key equivalent using the `NSEvent` method `charactersIgnoringModifiers` (page 1074).

Availability

Available in Mac OS X v10.0 and later.

See Also

`performKeyEquivalent:` (page 3195) (NSView)

`performKeyEquivalent:` (page 478) (NSButton)

Declared In

`NSResponder.h`

performMnemonic:

Overridden by subclasses to handle a mnemonic.

```
- (BOOL)performMnemonic:(NSString *)aString
```

Parameters

aString

A string containing mnemonic character code or codes.

Discussion

If the character code or codes in *aString* match the receiver's mnemonic, the receiver should perform the mnemonic and return `YES`. The default implementation does nothing and returns `NO`. Mnemonics are not supported in Mac OS X.

Availability

Available in Mac OS X v10.0 and later.

See Also

`performMnemonic:` (page 3195) (NSView)

Declared In

`NSResponder.h`

presentError:

Presents an error alert to the user as an application-modal dialog.

```
- (BOOL)presentError:(NSError *)anError
```

Parameters

anError

An object containing information about an error.

Discussion

The alert displays information found in the `NSError` object *anError*; this information can include error description, recovery suggestion, failure reason, and button titles (all localized). The method returns `YES` if error recovery succeeded and `NO` otherwise. For error recovery to be attempted, an recovery-attempter object (that is, an object conforming to the `NSErrorRecoveryAttempting` informal protocol) must be associated with *anError*.

The default implementation of this method sends `willPresentError:` (page 2205) to `self`. By doing this, `NSResponder` gives subclasses an opportunity to customize error presentation. It then forwards the message, passing any customized error object, to the next responder; if there is no next responder, it passes the error object to `NSApp`, which displays a document-modal error alert. When the user dismisses the alert, any recovery attempter associated with the error object is given a chance to recover from the error. See the class description for the precise route up the responder chain (plus document and controller objects) this message might travel.

It is not recommended that you attempt to override this method. If you wish to customize the error presentation, override `willPresentError:` (page 2205) instead.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 2187)

Related Sample Code

AbstractTree
Core Data HTML Store
CoreRecipes
DesktopImage
Image Kit with Core Data

Declared In

`NSResponder.h`

presentError:modalForWindow:delegate:didPresentSelector:contextInfo:

Presents an error alert to the user as a document-modal sheet attached to document window.

```
- (void)presentError:(NSError *)error modalForWindow:(NSWindow *)aWindow
    delegate:(id)delegate didPresentSelector:(SEL)didPresentSelector
    contextInfo:(void *)contextInfo
```

Parameters

error

The object encapsulating information about the error.

aWindow

The window object identifying the window owning the document-modal sheet.

delegate

The modal delegate for the sheet.

didPresentSelector

A selector identifying the message to be sent to the modal delegate. The *didPresentSelector* selector must have the signature:

```
- (void)didPresentErrorWithRecovery:(BOOL)didRecover
   contextInfo:(void *)contextInfo
```

contextInfo

Supplemental data to be passed to the modal delegate; can be `NULL`.

Discussion

The information displayed in the alert is extracted from the `NSError` object *error*; it may include a description, recovery suggestion, failure reason, and button titles (all localized). Once the user dismisses the alert and any recovery attempter associated with the error object has had a chance to recover from it, the receiver sends a message identified by *didPresentSelector* to the modal delegate *delegate*. (A recovery attempter is an object that conforms to the `NSErrorRecoveryAttempting` informal protocol.)

The modal delegate implements the method identified by *didPresentSelector* to perform any post-error processing if recovery failed or was not attempted (that is, *didRecover* is `NO`). Any supplemental data is passed to the modal delegate via *contextInfo*.

The default implementation of this method sends `willPresentError:` (page 2205) to `self`. By doing this, `NSResponder` gives subclasses an opportunity to customize error presentation. It then forwards the message, passing any customized error, to the next responder or; if there is no next responder, it passes the error object to `NSApp`, which displays a document-modal error alert. When the user dismisses the alert, any recovery attempter associated with the error object is given a chance to recover from the error. See the class description for the precise route up the responder chain (plus document and controller objects) this message might travel.

It is not recommended that you attempt to override this method. If you wish to customize the error presentation, override `willPresentError:` (page 2205) instead.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- presentError:` (page 2187)

Related Sample Code

Sketch+Accessibility

ZipBrowser

Declared In

`NSResponder.h`

resignFirstResponder

Notifies the receiver that it's been asked to relinquish its status as first responder in its window.

```
- (BOOL)resignFirstResponder
```

Discussion
The default implementation returns `YES`, resigning first responder status. Subclasses can override this method to update state or perform some action such as unhighlighting the selection, or to return `NO`, refusing to relinquish first responder status.

Use the `NSWindow` `makeFirstResponder:` (page 3344) method, not this method, to make an object the first responder. Never invoke this method directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `becomeFirstResponder` (page 2144)
- `acceptsFirstResponder` (page 2143)

Related Sample Code
From A View to A Movie
From A View to A Picture
NURBSSurfaceVertexProg
SurfaceVertexProgram
VertexPerformanceDemo

Declared In
`NSResponder.h`

rightMouseDown:

Informs the receiver that the user has pressed the right mouse button.

```
- (void)rightMouseDown:(NSEvent *)theEvent
```

Parameters
theEvent
An object encapsulating information about the mouse-down event.

Discussion
The default implementation simply passes this message to the next responder.

Note: The `NSView` implementation of this method does not pass the message up the responder chain, it handles it directly. See `rightMouseDown:` in *NSView Class Reference* for details.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

rightMouseDragged:

Informs the receiver that the user has moved the mouse with the right button pressed .

```
- (void)rightMouseDragged:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the mouse-dragged event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSResponder.h

rightMouseUp:

Informs the receiver that the user has released the right mouse button.

```
- (void)rightMouseUp:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the mouse-up event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSResponder.h

rotateWithEvent:

Informs the receiver that the user has begun a rotation gesture.

```
- (void)rotateWithEvent:(NSEvent *)event
```

Parameters

event

An event object representing the rotate gesture.

Discussion

The event will be sent to the view under the touch in the key window.

Availability

Available in Mac OS X v10.6 and later.

Declared In
NSResponder.h

scrollLineDown:

Implemented by subclasses to scroll the receiver one line down in its scroll view, without changing the selection.

```
- (void)scrollLineDown:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
NSResponder declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- scrollLineUp: (page 2191)
lineScroll (page 2350) (NSScrollView)

Declared In
NSResponder.h

scrollLineUp:

Implemented by subclasses to scroll the receiver one line up in its scroll view, without changing the selection.

```
- (void)scrollLineUp:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
NSResponder declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- scrollLineDown: (page 2191)
lineScroll (page 2350) (NSScrollView)

Declared In
NSResponder.h

scrollPageDown:

Implemented by subclasses to scroll the receiver one page down in its scroll view, without changing the selection.

```
- (void)scrollPageDown:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- pageDown:` (page 2184)

`- pageUp:` (page 2185)

`pageScroll` (page 2351) (NSScrollView)

Declared In

`NSResponder.h`

scrollPageUp:

Implemented by subclasses to scroll the receiver one page up in its scroll view, without changing the selection.

```
- (void)scrollPageUp:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- pageDown:` (page 2184)

`- pageUp:` (page 2185)

`pageScroll` (page 2351) (NSScrollView)

Declared In

`NSResponder.h`

scrollToBeginningOfDocument:

Implemented by subclasses to scroll the receiver to the beginning of the document, without changing the selection.

```
- (void)scrollToBeginningOfDocument:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

scrollToEndOfDocument:

Implemented by subclasses to scroll the receiver to the end of the document, without changing the selection.

```
- (void)scrollToEndOfDocument:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

scrollWheel:

Informs the receiver that the mouse's scroll wheel has moved.

```
- (void)scrollWheel:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the wheel-scrolling event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

selectAll:

Implemented by subclasses to select all selectable elements.

```
- (void)selectAll:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

selectLine:

Implemented by subclasses to select all elements in the line or lines containing the selection or insertion point.

```
- (void)selectLine:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

selectParagraph:

Implemented by subclasses to select all paragraphs containing the selection or insertion point.

```
- (void)selectParagraph:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

selectToMark:

Implemented by subclasses to select all items from the insertion point or selection to a previously placed mark, including the selection itself if not empty.

```
- (void)selectToMark:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
NSResponder declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setMark: (page 2196)
- deleteToMark: (page 2150)

Declared In
NSResponder.h

selectWord:

Implemented by subclasses to extend the selection to the nearest word boundaries outside it (up to, but not including, word delimiters).

```
- (void)selectWord:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
NSResponder declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

setInterfaceStyle:

Sets the receiver's style to the style specified by *interfaceStyle*, such as NSMacintoshInterfaceStyle or NSWindows95InterfaceStyle.

```
- (void)setInterfaceStyle:(NSInterfaceStyle)interfaceStyle
```

Parameters

interfaceStyle

An `enum` constant identifying an interface style.

Discussion

`setInterfaceStyle:` is an abstract method in `NSResponder`, but is overridden in classes such as `NSWindow` and `NSView` to actually set the interface style. You should almost never need to invoke or override this method, but if you do override it, your version should always invoke the implementation in `super`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- interfaceStyle` (page 2158)

Declared In

`NSInterfaceStyle.h`

setMark:

Implemented by subclasses to set a mark at the insertion point or selection, which is used by `deleteToMark:` (page 2150) and `selectToMark:` (page 2195).

```
- (void)setMark:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- swapWithMark:` (page 2198)

Declared In

`NSResponder.h`

setMenu:

Sets the receiver's menu.

```
- (void)setMenu:(NSMenu *)aMenu
```

Parameters

aMenu

The menu object to set as the receiver's menu.

Discussion
If the receiver is an `NSApplication` object, this method sets the main menu, typically set using `setMainMenu:` (page 170).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- menu` (page 2163)

Related Sample Code
From A View to A Movie
From A View to A Picture
VertexPerformanceTest

Declared In
`NSResponder.h`

setNextResponder:

Sets the receiver's next responder.

```
- (void)setNextResponder:(NSResponder *)aResponder
```

Parameters
aResponder
An object that inherits, directly or indirectly, from `NSResponder`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- nextResponder` (page 2182)

Declared In
`NSResponder.h`

shouldBeTreatedAsInkEvent:

Returns `YES` if the specified event should be treated as an ink event, `NO` if it should be treated as a mouse event.

```
- (BOOL)shouldBeTreatedAsInkEvent:(NSEvent *)theEvent
```

Parameters
theEvent
An event object representing the event to be tested.

Discussion
This method provides the ability to distinguish when a pen-down should start inking versus when a pen-down should be treated as a mouse down event. This allows for a write-anywhere model for pen-based input.

The default implementation in `NSApplication` sends the method to the `NSWindow` object under the pen. If the window is inactive, this method returns `YES`, unless the pen-down is in the window drag region. If the window is active, this method is sent to the `NSView` object under the pen.

The default implementation in `NSView` returns `YES`, and `NSControl` overrides and returns `NO`. This allows write-anywhere over most `NSView` objects, but allows the pen to be used to track in controls and to move windows.

A custom view should override this method to get the correct behavior for a pen-down in the view.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSResponder.h`

showContextHelp:

Implemented by subclasses to invoke the help system, displaying information relevant to the receiver and its current state.

```
- (void)showContextHelp:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- helpRequested:` (page 2153)

Declared In
`NSResponder.h`

swapWithMark:

Swaps the mark and the selection or insertion point, so that what was marked is now the selection or insertion point, and what was the insertion point or selection is now the mark.

```
- (void)swapWithMark:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMark:` (page 2196)

Declared In
`NSResponder.h`

swipeWithEvent:

Informs the receiver that the user has begun a swipe gesture.

```
- (void)swipeWithEvent:(NSEvent *)event
```

Parameters

event

An event object representing the swipe gesture.

Discussion
The event will be sent to the view under the touch in the key window.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSResponder.h`

tabletPoint:

Informs the receiver that a tablet-point event has occurred.

```
- (void)tabletPoint:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the tablet-point event.

Discussion
Tablet events are represented by `NSEvent` objects of type `NSTabletPoint` (page 1095). They describe the current state of a transducer (that is, a pointing device) that is in proximity to its tablet, reflecting changes such as location, pressure, tilt, and rotation. See the `NSEvent` reference for the methods that allow you to extract this and other information from *theEvent*. The default implementation of `NSResponder` passes the message to the next responder.

Availability
Available in Mac OS X v10.4 or later.

See Also
- `tabletProximity:` (page 2200)

Declared In
`NSResponder.h`

tabletProximity:

Informs the receiver that a tablet-proximity event has occurred.

```
- (void)tabletProximity:(NSEvent *)theEvent
```

Parameters

theEvent

An object encapsulating information about the tablet-point event.

Discussion

Tablet events are represented by `NSEvent` objects of type `NSTabletProximity` (page 1095). Tablet devices generate proximity events when the transducer (pointing device) nears a tablet and when it moves away from a tablet. From an event object of this type you can extract information about the kind of device and its capabilities, as well as the relation of this tablet-proximity event to various tablet-point events; see the `NSEvent` reference for details. The default implementation passes the message to the next responder.

Availability

Available in Mac OS X v10.4 or later.

See Also

`- tabletPoint:` (page 2199)

Declared In

`NSResponder.h`

touchesBeganWithEvent:

Informs the receiver that new set of touches has been recognized.

```
- (void)touchesBeganWithEvent:(NSEvent *)event
```

Parameters

event

An event object representing the beginning of a touch.

Discussion

The event will be sent to the view under the touch in the key window. To get the set of touches that began for this view (or descendants of this view) send [event touchesMatchingPhase:NSTouchPhaseBegan inView:self].

This is not always the point of contact with the touch device. A touch that transitions from resting to active may be part of a `touchesBeganWithEvent:` set.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSResponder.h`

touchesCancelledWithEvent:

Informs the receiver that tracking of touches has been cancelled for any reason..

```
- (void)touchesCancelledWithEvent:(NSEvent *)event
```

Parameters

event

An event object representing the cancellation of a touch event.

Discussion

The event will be sent to the view under the touch in the key window.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSResponder.h

touchesEndedWithEvent:

Informs the receiver that a set of touches have been removed.

```
- (void)touchesEndedWithEvent:(NSEvent *)event
```

Parameters

event

An event object representing the ending of a touch event.

Discussion

The event will be sent to the view under the touch in the key window. To get the set of touches that ended for this view (or descendants of this view) send [event touchesMatchingPhase:NSTouchPhaseEnded inView:self].

This is not always the point of removal with the touch device. A touch that transitions from active to resting may be part of an `touchesEndedWithEvent:` set.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSResponder.h

touchesMovedWithEvent:

Informs the receiver that one or more touches has moved.

```
- (void)touchesMovedWithEvent:(NSEvent *)event
```

Parameters

event

An event object representing a touch movement.

Discussion

The event will be sent to the view under the touch in the key window. To get the set of touches that moved for this view (or descendants of this view) send [event touchesMatchingPhase:NSTouchPhaseMoved inView:self].

Availability

Available in Mac OS X v10.6 and later.

Declared In
NSResponder.h

transpose:

Transposes the characters to either side of the insertion point and advances the insertion point past both of them. Does nothing to a selected range of text.

```
- (void)transpose:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
NSResponder declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

transposeWords:

Transposes the words to either side of the insertion point and advances the insertion point past both of them. Does nothing to a selected range of text.

```
- (void)transposeWords:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
NSResponder declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

tryToPerform:with:

Attempts to perform the action indicated method with a specified argument.

```
- (BOOL)tryToPerform:(SEL)anAction with:(id)anObject
```

Parameters

anAction
The selector identifying the action method.

anObject
The object to use as the sole argument of the action method.

Return Value
Returns `NO` if no responder is found that responds to *anAction*, `YES` otherwise.

Discussion
If the receiver responds to *anAction*, it invokes the method with *anObject* as the argument and returns `YES`. If the receiver doesn't respond, it sends this message to its next responder with the same selector and object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- doCommandBySelector:` (page 2152)
`sendAction:to:from:` (page 166) (NSApplication)

Declared In
`NSResponder.h`

undoManager

Returns the undo manager for this responder.

```
- (NSUndoManager *)undoManager
```

Discussion
`NSResponder`'s implementation simply passes this message to the next responder.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DotViewUndo
Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility
Sketch-112

Declared In
`NSResponder.h`

uppercaseWord:

Implemented by subclasses to make uppercase every letter in the word or words surrounding the insertion point or selection, expanding the selection if necessary.

```
- (void)uppercaseWord:(id)sender
```

Parameters
sender
Typically the object that invoked this method.

Discussion
If either end of the selection partially covers a word, that entire word is made uppercase. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lowercaseWord:` (page 2160)
- `capitalizeWord:` (page 2145)
- `changeCaseOfLetter:` (page 2146)

Declared In
`NSResponder.h`

validRequestorForSendType:returnType:

Overridden by subclasses to determine what services are available.

```
- (id)validRequestorForSendType:(NSString *)sendType returnType:(NSString
    *)returnType
```

Parameters

sendType
A string identifying the send type of pasteboard data. May be an empty string (see discussion).

returnType
A string identifying the return type of pasteboard data. May be an empty string (see discussion).

Return Value
If the receiver can place data of *sendType* on the pasteboard and receive data of *returnType*, it should return `self`; otherwise it should return either `[super validRequestorForSendType:returnType:]` or `[[self nextResponder] validRequestorForSendType:returnType:]`, which allows an object higher up in the responder chain to have an opportunity to handle the message.

Discussion
With each event, and for each service in the Services menu, the application object sends this message up the responder chain with the send and return type for the service being checked. This method is therefore invoked many times per event. The default implementation simply forwards this message to the next responder, ultimately returning `nil`.

Either *sendType* or *returnType*—but not both—may be empty. If *sendType* is empty, the service doesn't require input from the application requesting the service. If *returnType* is empty, the service doesn't return data.

Availability
Available in Mac OS X v10.0 and later.

See Also
`registerServicesMenuSendTypes:returnTypes:` (page 158) (`NSApplication`)
`writeSelectionToPasteboard:types:` (page 3798) (`NSServicesRequests` protocol)
`readSelectionFromPasteboard:` (page 3797) (`NSServicesRequests` protocol)

Declared In
NSResponder.h

willPresentError:

Implemented by subclasses to return a custom version of the supplied error object that is more suitable for presentation in alert sheets and dialogs.

```
- (NSError *)willPresentError:(NSError *)anError
```

Parameters

anError
The error object to be customized.

Return Value
The customized error object; if you decide not to customize the error presentation, return by sending this message to `super` (that is, `return [super willPresentError:anError]`).

Discussion
When overriding this method, you can examine *anError* and, if its localized description or recovery information is unhelpfully generic, return an error object with more specific localized text. If you do this, always use the domain and error code of the `NSError` object to distinguish between errors whose presentation you want to customize and those you do not. Don't make decisions based on the localized description, recovery suggestion, or recovery options because parsing localized text is problematic.

The default implementation of this method simply returns *anError* unchanged.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `presentError:` (page 2187)
- `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 2187)

Declared In
NSResponder.h

yank:

Replaces the insertion point or selection with text from the kill buffer.

```
- (void)yank:(id)sender
```

Parameters

sender
Typically the object that invoked this method.

Discussion
If invoked sequentially, cycles through the kill buffer in reverse order. See "Standard Action Methods for Selecting and Editing" in *Cocoa Event-Handling Guide* for more information on the kill buffer. `NSResponder` declares but doesn't implement this method.

Availability
Available in Mac OS X v10.0 and later.

See Also

- `deleteToBeginningOfLine:` (page 2149)
- `deleteToEndOfLine:` (page 2150)
- `deleteToBeginningOfParagraph:` (page 2149)
- `deleteToEndOfParagraph:` (page 2150)
- `deleteToMark:` (page 2150)

Declared In

`NSResponder.h`

NSRuleEditor Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSRuleEditor.h

Overview

An `NSRuleEditor` object is a view that allows the user to visually create and configure a list of options which are expressed by the rule editor as a predicate (see *Predicate Programming Guide*). The view has a delegate which offers a tree of choices to the view. The choices are presented by the view to the user as a row of popup buttons, static text fields, and custom views. Each row in the list represents a particular path down the tree of choices.

`NSRuleEditor` exposes one binding, `rows`. You can bind `rows` to an ordered collection (such as an instance of `NSMutableArray`). Each object in the collection should have the following properties:

@"rowType"
: An integer representing the type of the row (`NSRuleEditorRowType`).

@"subrows"
: An ordered to-many relation (such as an instance of `NSMutableArray`) containing the directly nested subrows for the given row.

@"displayValues"
: An ordered to-many relation containing the display values for the row.

@"criteria"
: An ordered to-many relation containing the criteria for the row.

Tasks

Configuring a Rule Editor

- `delegate` (page 2212)
  Returns the receiver's delegate.
- `setDelegate:` (page 2223)
  Sets the receiver's delegate.
- `isEditable` (page 2215)
  Returns a Boolean value that indicates whether the receiver is editable.
- `setEditable:` (page 2224)
  Sets whether the receiver is editable.
- `nestingMode` (page 2215)
  Returns the nesting mode for the receiver.
- `setNestingMode:` (page 2225)
  Sets the nesting mode for the receiver.
- `canRemoveAllRows` (page 2211)
  Returns a Boolean value that indicates whether all the rows can be removed.
- `setCanRemoveAllRows:` (page 2221)
  Sets whether all the rows can be removed.
- `rowHeight` (page 2219)
  Returns the row height for the receiver.
- `setRowHeight:` (page 2226)
  Sets the row height for the receiver.

Working with Formatting

- `formattingDictionary` (page 2213)
  Returns the formatting dictionary for the receiver.
- `setFormattingDictionary:` (page 2224)
  Sets the formatting dictionary for the receiver.
- `formattingStringsFilename` (page 2214)
  Returns the name of the strings file for the receiver.
- `setFormattingStringsFilename:` (page 2224)
  Sets the name of the strings file used for formatting.

Providing Data

- `reloadCriteria` (page 2217)
  Instructs the receiver to refetch criteria from its delegate.
- `setCriteria:andDisplayValues:forRowAtIndex:` (page 2222)
  Modifies the row at a given index to contain the given items and values.

- `criteriaForRow:` (page 2211)
  Returns the currently chosen items for a given row.
- `displayValuesForRow:` (page 2212)
  Returns the chosen values for a given row.

Obtaining Row Information

- `numberOfRows` (page 2215)
  Returns the number of rows in the receiver.
- `parentRowForRow:` (page 2216)
  Returns the index of the parent of a given row.
- `rowForDisplayValue:` (page 2219)
  Returns the index of the row containing a given value.
- `rowTypeForRow:` (page 2220)
  Returns the type of a given row.
- `subrowIndexesForRow:` (page 2227)
  Returns the immediate subrows of a given row.

Working with the Selection

- `selectedRowIndexes` (page 2221)
  Returns the indexes of the receiver's selected rows.
- `selectRowIndexes:byExtendingSelection:` (page 2221)
  Sets in the receiver the indexes of rows that are selected.

Manipulating Rows

- `addRow:` (page 2210)
  Adds a row to the receiver.
- `insertRowAtIndex:withType:asSubrowOfRow:animate:` (page 2214)
  Adds a new row of a given type at a given location.
- `removeRowAtIndex:` (page 2217)
  Removes the row at a given index.
- `removeRowsAtIndexes:includeSubrows:` (page 2218)
  Removes the rows at given indexes.

Working with Predicates

- `predicate` (page 2216)
  Returns the predicate for the receiver.
- `reloadPredicate` (page 2217)
  Instructs the receiver to regenerate its predicate by invoking the corresponding delegate method.

- `predicateForRow:` (page 2216)
  Returns the predicate for a given row.

Supporting Bindings

- `rowClass` (page 2218)
  Returns the class used to create a new row in the “rows” binding.
- `setRowClass:` (page 2225)
  Sets the class to use to create a new row in the "rows” binding.
- `rowTypeKeyPath` (page 2220)
  Returns the key path for the row type.
- `setRowTypeKeyPath:` (page 2226)
  Sets the key path for the row type.
- `subrowsKeyPath` (page 2227)
  The key path for the subrows.
- `setSubrowsKeyPath:` (page 2227)
  Set the key path for the subrows.
- `criteriaKeyPath` (page 2212)
  Returns the criteria key path.
- `setCriteriaKeyPath:` (page 2222)
  Sets the criteria key path.
- `displayValuesKeyPath` (page 2213)
  Returns the display values key path.
- `setDisplayValuesKeyPath:` (page 2223)
  Sets the display values key path.

Overriding ViewDidMoveToWindow

- `viewDidMoveToWindow` (page 2228)
  Overrides the `NSView` implementation.

Instance Methods

addRow:

Adds a row to the receiver.

```
- (void)addRow:(id)sender
```

Parameters

sender
  Typically the object that sent the message.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
PhotoSearch
PredicateEditorSample

Declared In
`NSRuleEditor.h`

canRemoveAllRows

Returns a Boolean value that indicates whether all the rows can be removed.

```
- (BOOL)canRemoveAllRows
```

Return Value
`YES` if all the rows can be removed, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setCanRemoveAllRows:` (page 2221)

Declared In
`NSRuleEditor.h`

criteriaForRow:

Returns the currently chosen items for a given row.

```
- (NSArray *)criteriaForRow:(NSInteger)row
```

Parameters

row
The index of a row in the receiver.

Return Value
The currently chosen items for row *row*.

Discussion
The items returned are the same as those returned from the delegate method "Configuring a Rule Editor" (page 2208).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

criteriaKeyPath

Returns the criteria key path.

```
- (NSString *)criteriaKeyPath
```

Return Value
The criteria key path.

Discussion
The default value is `@"criteria"`.

The key path is used to get the criteria for a row in the "rows" binding. The criteria objects are what the delegate returns from "Configuring a Rule Editor" (page 2208). The corresponding property should be an ordered to-many relationship.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setCriteriaKeyPath: (page 2222)

Declared In
NSRuleEditor.h

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setDelegate: (page 2223)

Declared In
NSRuleEditor.h

displayValuesForRow:

Returns the chosen values for a given row.

```
- (NSArray *)displayValuesForRow:(NSInteger)row
```

Parameters

row
The index of a row in the receiver.

Return Value
The chosen values (strings, views, or menu items) for row *row*.

Discussion
The values returned are the same as those returned from the delegate method "Working with Formatting" (page 2208).

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

displayValuesKeyPath

Returns the display values key path.

```
- (NSString *)displayValuesKeyPath
```

Return Value
The display values key path.

Discussion
The default is `@"displayValues"`.

The key path is used to get the display values for a row in the "rows" binding. The display values are what the delegate returns from "Working with Formatting" (page 2208). The corresponding property should be an ordered to-many relationship.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setDisplayValuesKeyPath: (page 2223)

Declared In
NSRuleEditor.h

formattingDictionary

Returns the formatting dictionary for the receiver.

```
- (NSDictionary *)formattingDictionary
```

Return Value
The formatting dictionary for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setFormattingStringsFilename: (page 2224)

Declared In
NSRuleEditor.h

formattingStringsFilename

Returns the name of the strings file for the receiver.

```
- (NSString *)formattingStringsFilename
```

Return Value
The name of the strings file for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setFormattingStringsFilename: (page 2224)

Declared In
NSRuleEditor.h

insertRowAtIndex:withType:asSubrowOfRow:animate:

Adds a new row of a given type at a given location.

```
- (void)insertRowAtIndex:(NSInteger)rowIndex withType:(NSRuleEditorRowType)rowType
    asSubrowOfRow:(NSInteger)parentRow animate:(BOOL)shouldAnimate
```

Parameters

rowIndex
The index at which the new row should be inserted. *rowIndex* must be greater than *parentRow*, and much specify a row that does not fall amongst the children of some other parent.

rowType
The type of the new row.

parentRow
The index of the row of which the new row is a child. Pass -1 to indicate that the new row should be a root row.

shouldAnimate
YES if creation of the new row should be animated, otherwise NO.

Special Considerations

Important: If *parentRow* is greater than or equal to *rowIndex*, or if *rowIndex* would fall amongst the children of some other parent, or if the nesting mode forbids this configuration, an NSInvalidArgumentException is raised.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

isEditable

Returns a Boolean value that indicates whether the receiver is editable.

```
- (BOOL)isEditable
```

Return Value
`YES` if the receiver is editable, otherwise `NO`.

Discussion
The default is `YES`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setEditable:` (page 2224)

Declared In
`NSRuleEditor.h`

nestingMode

Returns the nesting mode for the receiver.

```
- (NSRuleEditorNestingMode)nestingMode
```

Return Value
The nesting mode for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setNestingMode:` (page 2225)

Declared In
`NSRuleEditor.h`

numberOfRows

Returns the number of rows in the receiver.

```
- (NSInteger)numberOfRows
```

Return Value
The number of rows in the receiver.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
PhotoSearch
PredicateEditorSample

Declared In
`NSRuleEditor.h`

parentRowForRow:

Returns the index of the parent of a given row.

```
- (NSInteger)parentRowForRow:(NSInteger)rowIndex
```

Parameters

rowIndex
The index of a row in the receiver.

Important: Raises an `NSRangeException` if *rowIndex* is less than `0` or greater than or equal to the number of rows.

Return Value
The index of the parent of the row at *rowIndex*. If the row at *rowIndex* is a root row, returns `-1`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

predicate

Returns the predicate for the receiver.

```
- (NSPredicate *)predicate
```

Return Value
If the delegate implements NSRuleEditor (page 2207), the predicate for the receiver. If the delegate does implement NSRuleEditor, or if the delegate does not return enough parts to construct a full predicate, returns `nil`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

predicateForRow:

Returns the predicate for a given row.

```
- (NSPredicate *)predicateForRow:(NSInteger)row
```

Parameters

row

The index of a row in the receiver.

Return Value
The predicate for the row at *row*.

Discussion
You should rarely have a need to call this directly, but you can override this method in a subclass to perform specialized predicate handling for certain criteria or display values.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

reloadCriteria

Instructs the receiver to refetch criteria from its delegate.

```
- (void)reloadCriteria
```

Discussion
You can use this method to indicate that the available criteria may have changed and should be refetched from the delegate and the popups recalculated. If any item in a given row is "orphaned" (that is, is no longer reported as a child of its previous parent), its criteria and display values are set to valid choices.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

reloadPredicate

Instructs the receiver to regenerate its predicate by invoking the corresponding delegate method.

```
- (void)reloadPredicate
```

Discussion
You typically invoke this method because something has changed (for example, a view's value).

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

removeRowAtIndex:

Removes the row at a given index.

```
- (void)removeRowAtIndex:(NSInteger)rowIndex
```

Parameters

rowIndex

The index of a row in the receiver.

Important: Raises an `NSRangeException` if *rowIndex* is less than `0` or greater than or equal to the number of rows.

Discussion

Any subrows of the deleted row are adopted by the parent of the deleted row, or are made root rows.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSRuleEditor.h`

removeRowsAtIndexes:includeSubrows:

Removes the rows at given indexes.

```
- (void)removeRowsAtIndexes:(NSIndexSet *)rowIndexes
    includeSubrows:(BOOL)includeSubrows
```

Parameters

rowIndexes

Indexes of one or more rows in the receiver.

Important: Raises an `NSRangeException` if any index in *rowIndexes* is less than `0` or greater than or equal to the number of rows.

includeSubrows

If `YES`, then sub-rows of deleted rows are also deleted; if `NO`, then each sub-row is adopted by its first non-deleted ancestor, or becomes a root row.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSRuleEditor.h`

rowClass

Returns the class used to create a new row in the “rows” binding.

```
- (Class)rowClass
```

Return Value

The class used to create a new row in the "rows" binding.

Discussion
By default, this is `NSMutableDictionary`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setRowClass:` (page 2225)

Declared In
`NSRuleEditor.h`

rowForDisplayValue:

Returns the index of the row containing a given value.

```
- (NSInteger)rowForDisplayValue:(id)displayValue
```

Parameters
displayValue
The display value (string, view, or menu item) of an item in the receiver. This value must not be `nil`.

Important: Raises `NSInvalidArgumentException` if *displayValue* is `nil`.

Return Value
The index of the row containing *displayValue*, or `NSNotFound`.

Discussion
This method searches each row via pointer equality for the given display value, which may be present as an alternative in a popup menu for that row.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

rowHeight

Returns the row height for the receiver.

```
- (CGFloat)rowHeight
```

Return Value
The row height for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setRowHeight:` (page 2226)

Related Sample Code
PredicateEditorSample

Declared In
`NSRuleEditor.h`

rowTypeForRow:

Returns the type of a given row.

```
- (NSRuleEditorRowType)rowTypeForRow:(NSInteger)rowIndex
```

Parameters

rowIndex
The index of a row in the receiver.

Important: Raises an `NSRangeException` if *rowIndex* is less than `0` or greater than or equal to the number of rows.

Return Value
The type of the row at *rowIndex*.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

rowTypeKeyPath

Returns the key path for the row type.

```
- (NSString *)rowTypeKeyPath
```

Return Value
The key path for the row type.

Discussion
The default value is `@"rowType"`.

The key path is used to get the row type in the "rows" binding. The corresponding property should be a number that specifies an `NSRuleEditorRowType` value (see "Row Types" (page 2229)).

Availability
Available in Mac OS X v10.5 and later.

See Also
- setRowTypeKeyPath: (page 2226)

Declared In
`NSRuleEditor.h`

selectedRowIndexes

Returns the indexes of the receiver's selected rows.

```
- (NSIndexSet *)selectedRowIndexes
```

Return Value
The indexes of the receiver's selected rows.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

selectRowIndexes:byExtendingSelection:

Sets in the receiver the indexes of rows that are selected.

```
- (void)selectRowIndexes:(NSIndexSet *)indexes byExtendingSelection:(BOOL)extend
```

Parameters

indexes

The indexes of rows in the receiver to select.

Important: Raises an NSRangeException if any index in *rowIndexes* is less than 0 or greater than or equal to the number of rows.

extend

If NO, the selected rows are specified by *indexes*. If YES, the rows indicated by *indexes* are added to the collection of already selected rows, providing multiple selection.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

setCanRemoveAllRows:

Sets whether all the rows can be removed.

```
- (void)setCanRemoveAllRows:(BOOL)val
```

Parameters

val

YES if all the rows can be removed, otherwise NO.

Availability
Available in Mac OS X v10.5 and later.

See Also
- canRemoveAllRows (page 2211)

Declared In
NSRuleEditor.h

setCriteria:andDisplayValues:forRowAtIndex:

Modifies the row at a given index to contain the given items and values.

```
- (void)setCriteria:(NSArray *)criteria andDisplayValues:(NSArray *)values
    forRowAtIndex:(NSInteger)rowIndex
```

Parameters

criteria

The array of criteria for the row at *rowIndex*. Pass an empty array to force the receiver to query its delegate. This value must not be nil.

Important: Raises an NSInvalidArgumentException if *criteria* is nil.

values

The array of values for the row at *rowIndex*. Pass an empty array to force the receiver to query its delegate. This value must not be nil.

Important: Raises an NSInvalidArgumentException if *values* is nil.

rowIndex

The index of a row in the receiver.

Important: Raises an NSRangeException if *rowIndex* is equal to or larger than the number of rows, or less than 0.

Discussion
It is your responsibility to ensure that each item in the array is a child of the previous item, and that the first item is a root item for the row type. If the last item has child items, then the items array will be extended by querying the delegate for child items until a childless item is reached. If *values* contains fewer objects than the (possibly extended) criteria array, then the delegate is queried to construct the remaining display values. If you want the delegate to be queried for all the criteria or all the display values, pass empty arrays; do not pass nil.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

setCriteriaKeyPath:

Sets the criteria key path.

```
- (void)setCriteriaKeyPath:(NSString *)keyPath
```

Parameters

keyPath

The criteria key path.

Discussion

The criteria key path is described in `criteriaKeyPath` (page 2212).

Availability

Available in Mac OS X v10.5 and later.

See Also

- `criteriaKeyPath` (page 2212)

Declared In

`NSRuleEditor.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)delegate
```

Parameters

delegate

The delegate for the receiver.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `delegate` (page 2212)

Declared In

`NSRuleEditor.h`

setDisplayValuesKeyPath:

Sets the display values key path.

```
- (void)setDisplayValuesKeyPath:(NSString *)keyPath
```

Parameters

keyPath

The display values key path.

Discussion

The display values key path is described in `displayValuesKeyPath` (page 2213).

Availability

Available in Mac OS X v10.5 and later.

See Also

- `displayValuesKeyPath` (page 2213)

Declared In
NSRuleEditor.h

setEditable:

Sets whether the receiver is editable.

```
- (void)setEditable:(BOOL)editable
```

Parameters

editable
YES if the receiver is editable, otherwise NO.

Availability
Available in Mac OS X v10.5 and later.

See Also
- isEditable (page 2215)

Declared In
NSRuleEditor.h

setFormattingDictionary:

Sets the formatting dictionary for the receiver.

```
- (void)setFormattingDictionary:(NSDictionary *)dictionary
```

Parameters

dictionary
The formatting dictionary for the receiver.

Discussion
If you set the formatting dictionary with this method, it sets the current to formatting strings file name nil (see formattingStringsFilename (page 2214)).

Availability
Available in Mac OS X v10.5 and later.

See Also
- setFormattingStringsFilename: (page 2224)

Declared In
NSRuleEditor.h

setFormattingStringsFilename:

Sets the name of the strings file used for formatting.

```
- (void)setFormattingStringsFilename:(NSString *)stringsFilename
```

Parameters

stringsFilename

The name of the strings file for the receiver.

Discussion

`NSRuleEditor` looks for a strings file with the given name in the main bundle and (if appropriate) the bundle containing the nib file from which it was loaded. If it finds a strings file resource with the given name, `NSRuleEditor` loads it and sets it as the formatting dictionary for the receiver. You can obtain the resulting dictionary using `formattingDictionary` (page 2213)].

If you set the formatting dictionary with `setFormattingDictionary:` (page 2224), it sets the current to formatting strings file name `nil` (see `formattingStringsFilename` (page 2214)).

Availability

Available in Mac OS X v10.5 and later.

See Also

- `formattingDictionary` (page 2213)

Declared In

`NSRuleEditor.h`

setNestingMode:

Sets the nesting mode for the receiver.

```
- (void)setNestingMode:(NSRuleEditorNestingMode)mode
```

Parameters

mode

The nesting mode for the receiver.

Discussion

You typically set the nesting mode at view creation time and do not subsequently modify it. The default is `NSRuleEditorNestingModeCompound`.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `nestingMode` (page 2215)

Declared In

`NSRuleEditor.h`

setRowClass:

Sets the class to use to create a new row in the "rows" binding.

```
- (void)setRowClass:(Class)rowClass
```

Parameters

rowClass

The class to use to create a new row in the "rows" binding.

Availability
Available in Mac OS X v10.5 and later.

See Also
- rowClass (page 2218)

Declared In
NSRuleEditor.h

setRowHeight:

Sets the row height for the receiver.

```
- (void)setRowHeight:(CGFloat)height
```

Parameters

height

The row height for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- rowHeight (page 2219)

Declared In
NSRuleEditor.h

setRowTypeKeyPath:

Sets the key path for the row type.

```
- (void)setRowTypeKeyPath:(NSString *)keyPath
```

Parameters

keyPath

The key path for the row type.

Discussion
The row type key path is described in rowTypeKeyPath (page 2220).

Availability
Available in Mac OS X v10.5 and later.

See Also
- rowTypeKeyPath (page 2220)

Declared In
NSRuleEditor.h

setSubrowsKeyPath:

Set the key path for the subrows.

```
- (void)setSubrowsKeyPath:(NSString *)keyPath
```

Parameters

keyPath

The key path for the subrows.

Discussion

The subrows key path is described in `subrowsKeyPath` (page 2227).

Availability

Available in Mac OS X v10.5 and later.

See Also

- `subrowsKeyPath` (page 2227)

Declared In

`NSRuleEditor.h`

subrowIndexesForRow:

Returns the immediate subrows of a given row.

```
- (NSIndexSet *)subrowIndexesForRow:(NSInteger)rowIndex
```

Parameters

rowIndex

The index of a row in the receiver, or `-1` to get the top-level rows.

Important: Raises an `NSRangeException` if *rowIndex* is less than `-1` or greater than or equal to the number of rows.

Return Value

The immediate subrows of the row at *rowIndex*.

Discussion

Rows are numbered starting at `0`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSRuleEditor.h`

subrowsKeyPath

The key path for the subrows.

```
- (NSString *)subrowsKeyPath
```

Return Value
The key path for the subrows.

Discussion
The default value is `@"subrows"`.

The key path is used to get the nested rows in the “rows” binding. The corresponding property should be an ordered to-many relationship containing additional bound row objects.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setSubrowsKeyPath:` (page 2227)

Declared In
`NSRuleEditor.h`

viewDidMoveToWindow

Overrides the `NSView` implementation.

```
- (void)viewDidMoveToWindow
```

Special Considerations

If you override this method in a subclass, you must invoke super’s implementation.

Constants

NSRuleEditorNestingMode

Specifies a type for nesting modes.

```
typedef NSUInteger NSRuleEditorNestingMode;
```

Discussion
See “Nesting Modes” (page 2228) for possible values.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

Nesting Modes

These constants specify the nesting mode for the rule editor.

```
enum {
    NSRuleEditorNestingModeSingle,
    NSRuleEditorNestingModeList,
    NSRuleEditorNestingModeCompound,
    NSRuleEditorNestingModeSimple
};
```

Constants

`NSRuleEditorNestingModeSingle`

Only a single row is allowed.

Plus/minus buttons are not shown.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorNestingModeList`

Allows a single list, with no nesting and no compound rows.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorNestingModeCompound`

Unlimited nesting and compound rows.

This is the default.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorNestingModeSimple`

One compound row at the top with subrows beneath it, and no further nesting allowed.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

Declared In

`NSRuleEditor.h`

NSRuleEditorRowType

Specifies a type for row types.

```
typedef NSUInteger NSRuleEditorRowType;
```

Discussion

See "Row Types" (page 2229) for possible values.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSRuleEditor.h`

Row Types

Specify the type of a rule editor row.

```
enum {
    NSRuleEditorRowTypeSimple,
    NSRuleEditorRowTypeCompound
};
```

Constants

`NSRuleEditorRowTypeSimple`

Specifies a simple row.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorRowTypeCompound`

Specifies a compound row.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

Declared In

`NSRuleEditor.h`

Predicate Part Keys

These strings are used as keys to the dictionary returned from the optional delegate method NSRuleEditor (page 2207). To construct a valid predicate, the union of the dictionaries for each item in the row must contain the required parts.

```
APPKIT_EXTERN NSString * const NSRuleEditorPredicateLeftExpression;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateRightExpression;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateComparisonModifier;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateOptions;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateOperatorType;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateCustomSelector;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateCompoundType;
```

Constants

`NSRuleEditorPredicateLeftExpression`

The corresponding value is an `NSExpression` object representing the left expression in the predicate.

This value is required for a non-`nil` comparison predicate.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateRightExpression`

The corresponding value is an `NSExpression` object representing the right expression in the predicate.

This value is required for a non-`nil` comparison predicate.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateComparisonModifier`

The corresponding value is an `NSNumber` object representing a `NSComparisonPredicateModifier` constant the of the predicate.

This value is optional—if not specified, `NSDirectPredicateModifier` is assumed.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateOptions`
: The corresponding value is an `NSNumber` object representing a `NSComparisonPredicate_Options`bitfield.

  If no value is specified, `0` (no options) is assumed.

  Available in Mac OS X v10.5 and later.

  Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateOperatorType`
: The corresponding value is an `NSNumber` object representing a `NSPredicateOperatorType` constant.

  This value is required for a non-`nil` comparison predicate.

  Available in Mac OS X v10.5 and later.

  Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateCustomSelector`
: The corresponding value is an `NSString` object representing a custom selector.

  If specified, this overrides the operator type, options, and comparison modifier.

  Available in Mac OS X v10.5 and later.

  Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateCompoundType`
: The corresponding value is an `NSNumber` object representing a `Compound Predicate Types` constant.

  If specified, the other keys are ignored and the predicate for the row will be an `NSCompoundPredicate` predicate whose subpredicates are the predicates of the subrows of the given row.

  Available in Mac OS X v10.5 and later.

  Declared in `NSRuleEditor.h`.

Declared In
`NSRuleEditor.h`

Notifications

NSRuleEditorRowsDidChangeNotification

This notification is posted to the default notification center whenever the view's rows change.

The object is the rule editor; there is no `userInfo` object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

NSRulerMarker Class Objective-C Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSRulerMarker.h
Companion guide	Rulers and Paragraph Styles
Related sample code	Rulers Sketch-112

Overview

An NSRulerMarker object displays a symbol on an NSRulerView object, indicating a location for whatever graphics element it represents in the client of the ruler view (for example, a margin or tab setting, or the edges of a graphic on the page).

Adopted Protocols

NSCoding

- `encodeWithCoder:`
- `initWithCoder:`

NSCopying

- `copyWithZone:`

Tasks

Creating Instances

- `initWithRulerView:markerLocation:image:imageOrigin:` (page 2237)
  Initializes a newly allocated ruler marker, associating it with (but not adding it to) a specified ruler view and assigning the attributes provided.

Getting the Ruler View

- `ruler` (page 2240)
  Returns the receiver's ruler view.

Setting the Image

- `setImage:` (page 2240)
  Sets the receiver's image.
- `image` (page 2236)
  Returns the image displayed by the receiver.
- `setImageOrigin:` (page 2241)
  Sets the point in the receiver's image that is positioned at the receiver's location on the ruler view.
- `imageOrigin` (page 2236)
  Returns the point in the receiver's image positioned at the receiver's location on the ruler view.
- `imageRectInRuler` (page 2236)
  Returns the rectangle occupied by the receiver's image.
- `thicknessRequiredInRuler` (page 2243)
  Returns the amount of the receiver's image that's displayed above or to the left of the ruler view's baseline.

Setting Movability

- `setMovable:` (page 2242)
  Sets whether the user can move the receiver in its ruler view.
- `isMovable` (page 2238)
  Returns whether the user can move the receiver on its ruler view.
- `setRemovable:` (page 2242)
  Sets whether the user can remove the receiver from its ruler view.
- `isRemovable` (page 2239)
  Returns whether the user can remove the receiver from its ruler view.

Setting the Location

- `setMarkerLocation:` (page 2241)
  Sets the location of the receiver in the coordinate system of the ruler view's client view.
- `markerLocation` (page 2239)
  Returns the location of the receiver in the coordinate system of the ruler view's client view.

Setting the Represented Object

- `setRepresentedObject:` (page 2242)
  Sets the object the receiver represents.
- `representedObject` (page 2239)
  Returns the object the receiver represents.

Drawing and Event Handling

- `drawRect:` (page 2235)
  Draws the receiver's image that appears in the supplied rectangle.
- `isDragging` (page 2238)
  Returns whether the receiver is being dragged.
- `trackMouse:adding:` (page 2243)
  Handles user manipulation of the receiver in its ruler view.

Instance Methods

drawRect:

Draws the receiver's image that appears in the supplied rectangle.

```
- (void)drawRect:(NSRect)aRect
```

Parameters

aRect
  The rectangle to be drawn, in the ruler view's coordinate system.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `imageRectInRuler` (page 2236)

Declared In
`NSRulerMarker.h`

image

Returns the image displayed by the receiver.

```
- (NSImage *)image
```

Return Value
The image displayed by the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setImage:` (page 2240)

Declared In
`NSRulerMarker.h`

imageOrigin

Returns the point in the receiver's image positioned at the receiver's location on the ruler view.

```
- (NSPoint)imageOrigin
```

Return Value
The point in the receiver's image positioned at the receiver's location on the ruler view, expressed in the image's coordinate system.

Discussion
For a horizontal ruler, the x coordinate of the image origin is aligned with the location of the marker, and the y coordinate lies on the baseline of the ruler. For vertical rulers, the y coordinate of the image origin is the location, and the x coordinate lies on the baseline.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setImageOrigin:` (page 2241)
- `imageRectInRuler` (page 2236)

Declared In
`NSRulerMarker.h`

imageRectInRuler

Returns the rectangle occupied by the receiver's image.

```
- (NSRect)imageRectInRuler
```

Return Value
The rectangle occupied by the receiver's image, in the ruler view's coordinate system, accounting for whether the ruler view's coordinate system is flipped.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawRect:` (page 2235)
- `thicknessRequiredInRuler` (page 2243)

Declared In
`NSRulerMarker.h`

initWithRulerView:markerLocation:image:imageOrigin:

Initializes a newly allocated ruler marker, associating it with (but not adding it to) a specified ruler view and assigning the attributes provided.

```
- (id)initWithRulerView:(NSRulerView *)aRulerView markerLocation:(CGFloat)location
    image:(NSImage *)anImage imageOrigin:(NSPoint)imageOrigin
```

Parameters

aRulerView
The ruler view with which to associate the ruler marker. This method raises an `NSInvalidArgumentException` if *aRulerView* is `nil`.

location
The x or y position of the marker in the client view's coordinate system, depending on whether the ruler view is horizontal or vertical.

anImage
The image displayed at the marker location. This method raises an `NSInvalidArgumentException` if *anImage* is `nil`.

imageOrigin
The point within the image positioned at the marker location, expressed in pixels relative to the lower-left corner of the image.

Return Value
An initialized ruler marker object.

Discussion
The image used to draw the marker must be appropriate for the orientation of the ruler. Markers may need to look different on a horizontal ruler than on a vertical ruler, and the ruler view neither scales nor rotates the images.

To add the new ruler marker to *aRulerView*, use either of NSRulerView's `addMarker:` (page 2251) or `trackMarker:withMouseEvent:` (page 2262) methods. `addMarker:` (page 2251) immediately puts the marker on the ruler, while `trackMarker:withMouseEvent:` (page 2262) allows the client view to intercede in the addition and placement of the marker.

A new ruler marker can be moved on its ruler view, but not removed. Use `setMovable:` (page 2242) and `setRemovable:` (page 2242) to change these attributes. The new ruler marker also has no represented object; use `setRepresentedObject:` (page 2242) to set one.

This method is the designated initializer for the NSRulerMarker class.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMarkerLocation:` (page 2241)
- `setImage:` (page 2240)
- `setImageOrigin:` (page 2241)

Related Sample Code
Rulers

Declared In
`NSRulerMarker.h`

isDragging

Returns whether the receiver is being dragged.

```
- (BOOL)isDragging
```

Return Value
`YES` if the receiver is being dragged, `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `trackMouse:adding:` (page 2243)

Declared In
`NSRulerMarker.h`

isMovable

Returns whether the user can move the receiver on its ruler view.

```
- (BOOL)isMovable
```

Return Value
`YES` if the user can move the receiver on its ruler view, `NO` otherwise.

Discussion
By default ruler markers are movable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMovable:` (page 2242)
- `isRemovable` (page 2239)

Declared In
`NSRulerMarker.h`

isRemovable

Returns whether the user can remove the receiver from its ruler view.

```
- (BOOL)isRemovable
```

Return Value
`YES` if the user can remove the receiver from its ruler view, `NO` otherwise.

Discussion
By default ruler markers are not removable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setRemovable:` (page 2242)
- `isMovable` (page 2238)

Declared In
`NSRulerMarker.h`

markerLocation

Returns the location of the receiver in the coordinate system of the ruler view's client view.

```
- (CGFloat)markerLocation
```

Return Value
The location of the receiver in the coordinate system of the ruler view's client view. This is an x position for a horizontal ruler, a y position for a vertical ruler.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMarkerLocation:` (page 2241)

Related Sample Code
Rulers

Declared In
`NSRulerMarker.h`

representedObject

Returns the object the receiver represents.

```
- (id < NSCopying >)representedObject
```

Return Value
The object the receiver represents.

Discussion
See About Ruler Markers for more information on the represented object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setRepresentedObject:` (page 2242)

Declared In
`NSRulerMarker.h`

ruler

Returns the receiver's ruler view.

```
- (NSRulerView *)ruler
```

Return Value
The receiver's ruler view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addMarker:` (page 2251) (`NSRulerView`)

Declared In
`NSRulerMarker.h`

setImage:

Sets the receiver's image.

```
- (void)setImage:(NSImage *)anImage
```

Parameters
anImage
The new image.

Discussion
The image used to draw the marker must be appropriate for the orientation of the ruler. Markers may need to look different on a horizontal ruler than on a vertical ruler, and the ruler view neither scales nor rotates the images.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `image` (page 2236)
- `setImageOrigin:` (page 2241)

Declared In
NSRulerMarker.h

setImageOrigin:

Sets the point in the receiver's image that is positioned at the receiver's location on the ruler view.

```
- (void)setImageOrigin:(NSPoint)aPoint
```

Parameters

aPoint

The point within the image positioned at the marker location, expressed in pixels relative to the lower-left corner of the image.

Discussion
For a horizontal ruler, the x coordinate of the image origin is aligned with the location of the marker, and the y coordinate lies on the baseline of the ruler. For vertical rulers, the y coordinate of the image origin is the location, and the x coordinate lies on the baseline.

Availability
Available in Mac OS X v10.0 and later.

See Also
- imageOrigin (page 2236)
- setImage: (page 2240)
- setMarkerLocation: (page 2241)

Declared In
NSRulerMarker.h

setMarkerLocation:

Sets the location of the receiver in the coordinate system of the ruler view's client view.

```
- (void)setMarkerLocation:(CGFloat)location
```

Parameters

location

The location of the receiver in the coordinate system of the ruler view's client view. This is an x position for a horizontal ruler, a y position for a vertical ruler.

Availability
Available in Mac OS X v10.0 and later.

See Also
- markerLocation (page 2239)
- setImageOrigin: (page 2241)

Related Sample Code
Rulers

Declared In
NSRulerMarker.h

setMovable:

Sets whether the user can move the receiver in its ruler view.

```
- (void)setMovable:(BOOL)flag
```

Parameters

flag

YES to allow the user to drag the marker image in the ruler, NO to make it immobile.

Discussion
By default ruler markers are movable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isMovable (page 2238)
- setRemovable: (page 2242)

Declared In
NSRulerMarker.h

setRemovable:

Sets whether the user can remove the receiver from its ruler view.

```
- (void)setRemovable:(BOOL)flag
```

Parameters

flag

YES to allow the user to drag the marker image off of the ruler and remove the marker, NO to prevent the user from removing the marker.

Discussion
By default ruler markers are not removable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isRemovable (page 2239)
- setMovable: (page 2242)

Related Sample Code
Rulers

Declared In
NSRulerMarker.h

setRepresentedObject:

Sets the object the receiver represents.

```
- (void)setRepresentedObject:(id < NSCopying >)anObject
```

Parameters

anObject

The new represented object.

Discussion

See About Ruler Markers for more information on the represented object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- representedObject (page 2239)

Related Sample Code

Rulers

Declared In

NSRulerMarker.h

thicknessRequiredInRuler

Returns the amount of the receiver's image that's displayed above or to the left of the ruler view's baseline.

```
- (CGFloat)thicknessRequiredInRuler
```

Return Value

The amount of the receiver's image that's displayed above or to the left of the ruler view's baseline, the height for a horizontal ruler or width for a vertical ruler.

Availability

Available in Mac OS X v10.0 and later.

See Also

- imageOrigin (page 2236)

Declared In

NSRulerMarker.h

trackMouse:adding:

Handles user manipulation of the receiver in its ruler view.

```
- (BOOL)trackMouse:(NSEvent *)theEvent adding:(BOOL)flag
```

Parameters

theEvent

The event that represents the user manipulation being attempted on the ruler marker.

flag

YES to indicate that the receiver is a new marker being added to its ruler view, NO otherwise.

Return Value
YES if the user manipulation was allowed, NO if it was not allowed.

Discussion
NSRulerView objects invoke this method automatically to add a new marker or to move or remove an existing marker. You should never need to invoke it directly.

If the receiver is a new marker being added to its ruler view (*flag* is YES), the receiver queries the ruler view's client before adding itself to the ruler view. If the client responds to rulerView:shouldAddMarker: (page 2265) and that method returns NO, this method immediately returns NO, and the new marker isn't added.

If the receiver is not a new marker being added to its ruler view (*flag* is NO), this method attempts to move or remove an existing marker, once again based on responses from the ruler view's client view. If the receiver is neither movable nor removable, this method immediately returns NO. Further, if the ruler view's client responds to rulerView:shouldMoveMarker: (page 2265) and returns NO, this method returns NO, indicating the receiver can't be moved.

If the receiver is being added or moved, this method queries the client view using rulerView:willAddMarker:atLocation: (page 2266) or rulerView:willMoveMarker:toLocation: (page 2266), respectively. If the client responds to the method, the return value is used as the receiver's location. These methods are invoked repeatedly as the receiver is dragged within the ruler view.

If the receiver is an existing marker being removed (dragged off the ruler), this method queries the client view using rulerView:shouldRemoveMarker: (page 2265). If the client responds to this method and returns NO, the marker is pinned to the ruler view's baseline (following the cursor on the baseline if it's movable).

When the user releases the mouse button, this method informs the client view of the marker's new status using rulerView:didAddMarker: (page 2263), rulerView:didMoveMarker: (page 2263), or rulerView:didRemoveMarker: (page 2264) as appropriate. The client view can use this notification to set the marker's represented object, modify its state and redisplay (for example, adjusting text layout around a new tab stop), or take whatever other action it might need.

If *flag* is YES and the user dragged the new marker away from the ruler, the marker isn't added, no message is sent, and this method returns NO.

See *Rulers and Paragraph Styles* for more information on these client methods.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isMovable (page 2238)
- isRemovable (page 2239)

Declared In
NSRulerMarker.h

NSRulerView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSRulerView.h
Companion guide	Rulers and Paragraph Styles
Related sample code	Rulers Sketch+Accessibility Sketch-112 WhackedTV

Class at a Glance

An NSRulerView displays a ruler and markers above or to the side of an NSScrollView's document view. Views within the NSScrollView can become clients of the ruler view, having it display markers for their elements, and receiving messages from the ruler view when the user manipulates the markers.

Principal Attributes

- Displays markers that represent elements of the client view.
- Displays in arbitrary units.
- Provides for an accessory view containing extra controls.

`setHasHorizontalRuler:` (page 2356) (`NSScrollView`)
`setHasVerticalRuler:` (page 2357) (`NSScrollView`)
`initWithScrollView:orientation:` (page 2253) Designated initializer.

Commonly Used Methods

`setClientView:` (page 2258)
Changes the ruler's client view.

`setMarkers:` (page 2259)
Sets the markers displayed by the ruler view.

`setAccessoryView:` (page 2258)
Sets the accessory view.

`trackMarker:withMouseEvent:` (page 2262)
Allows the user to add a new marker.

Overview

An NSRulerView resides in an NSScrollView, displaying a labeled ruler and markers for its client, the document view of the NSScrollView, or a subview of the document view.

Tasks

Creating Instances

`- initWithScrollView:orientation:` (page 2253)
Initializes a newly allocated NSRulerView to have *orientation* (`NSHorizontalRuler` or `NSVerticalRuler`) within *aScrollView*.

Altering Measurement Units

`+ registerUnitWithName:abbreviation:unitToPointsConversionFactor:stepUpCycle:stepDownCycle:` (page 2250)
Registers a new unit of measurement with the NSRulerView class, making it available to all instances of NSRulerView.

`- setMeasurementUnits:` (page 2259)
Sets the measurement units used by the ruler to *unitName*.

`- measurementUnits` (page 2254)
Returns the full name of the measurement units in effect for the receiver.

Setting the Client View

`- setClientView:` (page 2258)
Sets the receiver's client view to *aView*, without retaining it, and removes its ruler markers, after informing the prior client of the change using `rulerView:willSetClientView:` (page 2267).

- `clientView` (page 2252)
  Returns the receiver's client view, if it has one.

Setting an Accessory View

- `setAccessoryView:` (page 2258)
  Sets the receiver's accessory view to *aView*.
- `accessoryView` (page 2251)
  Returns the receiver's accessory view, if it has one.

Setting the Zero Mark Position

- `setOriginOffset:` (page 2260)
  Sets the distance to the zero hash mark from the bounds origin of the NSScrollView's document view (not of the receiver's client view), in the document view's coordinate system.
- `originOffset` (page 2256)
  Returns the distance from the receiver's zero hash mark to the bounds origin of the NSScrollView's document view (not the receiver's client view), in the document view's coordinate system.

Adding and Removing Markers

- `setMarkers:` (page 2259)
  Sets the receiver's ruler markers to *markers*, removing any existing ruler markers and not consulting with the client view about the new markers.
- `markers` (page 2254)
  Returns the receiver's NSRulerMarkers.
- `addMarker:` (page 2251)
  Adds *aMarker* to the receiver, without consulting the client view for approval.
- `removeMarker:` (page 2256)
  Removes *aMarker* from the receiver, without consulting the client view for approval.
- `trackMarker:withMouseEvent:` (page 2262)
  Tracks the mouse to add *aMarker* based on the initial mouse-down or mouse-dragged event *theEvent*.

Drawing Temporary Ruler Lines

- `moveRulerlineFromLocation:toLocation:` (page 2255)
  Draws temporary lines in the ruler area.

Drawing

- `drawHashMarksAndLabelsInRect:` (page 2252)
  Draws the receiver's hash marks and labels in *aRect*, which is expressed in the receiver's coordinate system.
- `drawMarkersInRect:` (page 2252)
  Draws the receiver's markers in *aRect*, which is expressed in the receiver's coordinate system.
- `invalidateHashMarks` (page 2253)
  Forces recalculation of the hash mark spacing for the next time the receiver is displayed.

Ruler Layout

- `setScrollView:` (page 2262)
  Sets the NSScrollView that owns the receiver to *scrollView*, without retaining it.
- `scrollView` (page 2258)
  Returns the NSScrollView object that contains the receiver.
- `setOrientation:` (page 2260)
  Sets the orientation of the receiver to *orientation*.
- `orientation` (page 2255)
  Returns the orientation of the receiver.
- `setReservedThicknessForAccessoryView:` (page 2260)
  Sets the room available for the receiver's accessory view to *thickness*.
- `reservedThicknessForAccessoryView` (page 2257)
  Returns the thickness reserved to contain the receiver's accessory view, its height or width depending on the receiver's orientation.
- `setReservedThicknessForMarkers:` (page 2261)
  Sets the room available for ruler markers to *thickness*.
- `reservedThicknessForMarkers` (page 2257)
  Returns the thickness reserved to contain the images of the receiver's ruler markers, the height or width depending on the receiver's orientation.
- `setRuleThickness:` (page 2262)
  Sets to *thickness* the thickness of the area where ruler hash marks and labels are drawn.
- `ruleThickness` (page 2257)
  Returns the thickness of the receiver's ruler area (the area where hash marks and labels are drawn), its height or width depending on the receiver's orientation.
- `requiredThickness` (page 2256)
  Returns the thickness needed for proper tiling of the receiver within an NSScrollView.
- `baselineLocation` (page 2251)
  Returns the location of the receiver's baseline, in its own coordinate system.
- `isFlipped` (page 2254) Deprecated in Mac OS X v10.6
  Returns `YES` if the NSRulerView's coordinate system is flipped, `NO` otherwise.

Adding markers

- `rulerView:shouldAddMarker:` (page 2265) *delegate method*
  Requests permission for `aRulerView` to add `aMarker`, an NSRulerMarker being dragged onto the ruler by the user.
- `rulerView:willAddMarker:atLocation:` (page 2266) *delegate method*
  Informs the client that `aRulerView` will add the new NSRulerMarker, `aMarker`.
- `rulerView:didAddMarker:` (page 2263) *delegate method*
  Informs the client that `aRulerView` allowed the user to add `aMarker`.

Moving markers

- `rulerView:shouldMoveMarker:` (page 2265) *delegate method*
  Requests permission for `aRulerView` to move `aMarker`.
- `rulerView:willMoveMarker:toLocation:` (page 2266) *delegate method*
  Informs the client that `aRulerView` will move `aMarker`, an NSRulerMarker already on the ruler view.
- `rulerView:didMoveMarker:` (page 2263) *delegate method*
  Informs the client that `aRulerView` allowed the user to move `aMarker`.

Removing markers

- `rulerView:shouldRemoveMarker:` (page 2265) *delegate method*
  Requests permission for `aRulerView` to remove `aMarker`.
- `rulerView:didRemoveMarker:` (page 2264) *delegate method*
  Informs the client that `aRulerView` allowed the user to remove `aMarker`.

Handling mouse events

- `rulerView:handleMouseDown:` (page 2264) *delegate method*
  Informs the client that the user has pressed the mouse button while the cursor is in the ruler area of `aRulerView`.

Changing client view

- `rulerView:willSetClientView:` (page 2267) *delegate method*
  Informs the client view that `aRulerView` is about to be appropriated by `newClient`.

Class Methods

registerUnitWithName:abbreviation:unitToPointsConversionFactor:stepUpCycle:stepDownCycle:

Registers a new unit of measurement with the NSRulerView class, making it available to all instances of NSRulerView.

```
+ (void)registerUnitWithName:(NSString *)unitName abbreviation:(NSString
    *)abbreviation unitToPointsConversionFactor:(CGFloat)conversionFactor
    stepUpCycle:(NSArray *)stepUpCycle stepDownCycle:(NSArray *)stepDownCycle
```

Discussion
unitName is the name of the unit in English, in plural form and capitalized by convention—"Inches", for example. The unit name is used as a key to identify the measurement units and so shouldn't be localized. *abbreviation* is a localized short form of the unit name, such as "in" for Inches. *conversionFactor* is the number of PostScript points in the specified unit; there are 72.0 points per inch, for example. *stepUpCycle* and *stepDownCycle* are arrays of NSNumbers that specify how hash marks are calculated, as explained in "Setting Up a Ruler View". All numbers in *stepUpCycle* should be greater than 1.0, those in *stepDownCycle* should be less than 1.0.

NSRulerView supports these units by default:

Unit Name	Abbreviation	Points/Unit	Step-Up Cycle	Step-Down Cycle
Inches	in	72.0	2.0	0.5
Centimeters	cm	28.35	2.0	0.5, 0.2
Points	pt	1.0	10.0	0.5
Picas	pc	12.0	10.0	0.5

Availability
Available in Mac OS X v10.0 and later.

See Also
- setMeasurementUnits: (page 2259)

Related Sample Code
Rulers

Declared In
NSRulerView.h

Instance Methods

accessoryView

Returns the receiver's accessory view, if it has one.

```
- (NSView *)accessoryView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAccessoryView:` (page 2258)
- `reservedThicknessForAccessoryView` (page 2257)

Declared In
`NSRulerView.h`

addMarker:

Adds *aMarker* to the receiver, without consulting the client view for approval.

```
- (void)addMarker:(NSRulerMarker *)aMarker
```

Discussion
Raises an `NSInternalInconsistencyException` if the receiver has no client view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMarkers:` (page 2259)
- `removeMarker:` (page 2256)
- `markers` (page 2254)
- `trackMarker:withMouseEvent:` (page 2262)

Declared In
`NSRulerView.h`

baselineLocation

Returns the location of the receiver's baseline, in its own coordinate system.

```
- (CGFloat)baselineLocation
```

Discussion
This is a y position for horizontal rulers and an x position for vertical ones.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `ruleThickness` (page 2257)

Declared In
`NSRulerView.h`

clientView

Returns the receiver's client view, if it has one.

```
- (NSView *)clientView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setClientView:` (page 2258)

Declared In
`NSRulerView.h`

drawHashMarksAndLabelsInRect:

Draws the receiver's hash marks and labels in *aRect*, which is expressed in the receiver's coordinate system.

```
- (void)drawHashMarksAndLabelsInRect:(NSRect)aRect
```

Discussion
This method is invoked by `drawRect:` (page 2235)—you should never need to invoke it directly. You can define custom measurement units using the class method `registerUnitWithName:abbreviation:unitToPointsConversionFactor:stepUpCycle:stepDownCycle:` (page 2250). Override this method if you want to customize the appearance of the hash marks themselves.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `invalidateHashMarks` (page 2253)
- `drawMarkersInRect:` (page 2252)

Declared In
`NSRulerView.h`

drawMarkersInRect:

Draws the receiver's markers in *aRect*, which is expressed in the receiver's coordinate system.

```
- (void)drawMarkersInRect:(NSRect)aRect
```

Discussion
This method is invoked by `drawRect:` (page 2235); you should never need to invoke it directly, but you might want to override it if you want to do something different when drawing markers.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `reservedThicknessForMarkers` (page 2257)
- `drawHashMarksAndLabelsInRect:` (page 2252)

Declared In
`NSRulerView.h`

initWithScrollView:orientation:

Initializes a newly allocated NSRulerView to have *`orientation`* (`NSHorizontalRuler` or `NSVerticalRuler`) within *`aScrollView`*.

```
- (id)initWithScrollView:(NSScrollView *)aScrollView
    orientation:(NSRulerOrientation)orientation
```

Discussion
The new ruler view displays the user's preferred measurement units and has no client, markers, or accessory view. Unlike most subclasses of NSView, no initial frame rectangle is given for NSRulerView; its containing NSScrollView adjusts its frame rectangle as needed.

This method is the designated initializer for the NSRulerView class. Returns an initialized object.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSRulerView.h`

invalidateHashMarks

Forces recalculation of the hash mark spacing for the next time the receiver is displayed.

```
- (void)invalidateHashMarks
```

Discussion
You should never need to invoke this method directly, but might need to override it if you override `drawHashMarksAndLabelsInRect:` (page 2252).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawHashMarksAndLabelsInRect:` (page 2252)

Declared In
`NSRulerView.h`

isFlipped

Returns `YES` if the NSRulerView's coordinate system is flipped, `NO` otherwise.

```
- (BOOL)isFlipped
```

Discussion
A vertical ruler takes into account whether the coordinate system of the NSScrollView's document view—not the receiver's client view—is flipped. A horizontal ruler is always flipped.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
WhackedTV

Declared In
`NSRulerView.h`

markers

Returns the receiver's NSRulerMarkers.

```
- (NSArray *)markers
```

Discussion
The markers aren't guaranteed to be sorted in any particular order.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMarkers:` (page 2259)
- `addMarker:` (page 2251)
- `removeMarker:` (page 2256)
- `markerLocation` (page 2239) (NSRulerMarker)

Related Sample Code
Rulers

Declared In
`NSRulerView.h`

measurementUnits

Returns the full name of the measurement units in effect for the receiver.

```
- (NSString *)measurementUnits
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMeasurementUnits:` (page 2259)
+ `registerUnitWithName:abbreviation:unitToPointsConversionFactor:stepUpCycle:stepDownCycle:` (page 2250)

Declared In
`NSRulerView.h`

moveRulerlineFromLocation:toLocation:

Draws temporary lines in the ruler area.

```
- (void)moveRulerlineFromLocation:(CGFloat)oldLoc toLocation:(CGFloat)newLoc
```

Discussion
If *oldLoc* is 0 or greater, erases the ruler line at that location; if *newLoc* is 0 or greater, draws a new rulerline at that location. *oldLoc* and *newLoc* are expressed in the coordinate system of the NSRulerView, not the client or document view, and are x coordinates for horizontal rulers and y coordinates for vertical rulers. Use NSView's `convert...` methods to convert coordinates from the client or document view's coordinate system to that of the NSRulerView.

This method is useful for drawing highlight lines in the ruler to show the position or extent of an object while it's being dragged in the client view. The sender is responsible for keeping track of the number and positions of temporary lines—the NSRulerView only does the drawing.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Rulers
Sketch+Accessibility
Sketch-112

Declared In
`NSRulerView.h`

orientation

Returns the orientation of the receiver.

```
- (NSRulerOrientation)orientation
```

Discussion
Possible values are described in "Constants" (page 2267).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setOrientation:` (page 2260)

Declared In
`NSRulerView.h`

originOffset

Returns the distance from the receiver's zero hash mark to the bounds origin of the NSScrollView's document view (not the receiver's client view), in the document view's coordinate system.

```
- (CGFloat)originOffset
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setOriginOffset:` (page 2260)

Declared In
`NSRulerView.h`

removeMarker:

Removes *aMarker* from the receiver, without consulting the client view for approval.

```
- (void)removeMarker:(NSRulerMarker *)aMarker
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMarkers:` (page 2259)
- `addMarker:` (page 2251)

Declared In
`NSRulerView.h`

requiredThickness

Returns the thickness needed for proper tiling of the receiver within an NSScrollView.

```
- (CGFloat)requiredThickness
```

Discussion
This thickness is the height of a horizontal ruler and the width of a vertical ruler. The required thickness is the sum of the thicknesses of the ruler area, the marker area, and the accessory view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `ruleThickness` (page 2257)
- `reservedThicknessForMarkers` (page 2257)
- `reservedThicknessForAccessoryView` (page 2257)

Declared In
`NSRulerView.h`

reservedThicknessForAccessoryView

Returns the thickness reserved to contain the receiver's accessory view, its height or width depending on the receiver's orientation.

```
- (CGFloat)reservedThicknessForAccessoryView
```

Discussion
This thickness is automatically enlarged as necessary to the accessory view's thickness (but never automatically reduced). To prevent retiling of a ruler view's scroll view, you should set its maximal thickness upon creating using `setReservedThicknessForAccessoryView:` (page 2260).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSRulerView.h`

reservedThicknessForMarkers

Returns the thickness reserved to contain the images of the receiver's ruler markers, the height or width depending on the receiver's orientation.

```
- (CGFloat)reservedThicknessForMarkers
```

Discussion
This thickness is automatically enlarged as necessary to accommodate the thickest ruler marker image (but never automatically reduced). To prevent retiling of a ruler view's scroll view, you should set its maximal thickness upon creating using `setReservedThicknessForMarkers:` (page 2261).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- thicknessRequiredInRuler` (page 2243) (NSRulerMarker)

Declared In
`NSRulerView.h`

ruleThickness

Returns the thickness of the receiver's ruler area (the area where hash marks and labels are drawn), its height or width depending on the receiver's orientation.

```
- (CGFloat)ruleThickness
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setRuleThickness:` (page 2262)

Declared In
`NSRulerView.h`

scrollView

Returns the NSScrollView object that contains the receiver.

```
- (NSScrollView *)scrollView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setScrollView:` (page 2262)
- `setHorizontalRulerView:` (page 2359) (NSScrollView)
- `setVerticalRulerView:` (page 2363) (NSScrollView)

Declared In
`NSRulerView.h`

setAccessoryView:

Sets the receiver's accessory view to *aView*.

```
- (void)setAccessoryView:(NSView *)aView
```

Discussion
Raises an `NSInternalInconsistencyException` if *aView* is not `nil` and the receiver has no client view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `accessoryView` (page 2251)
- `reservedThicknessForAccessoryView` (page 2257)

Related Sample Code
Sketch+Accessibility

Declared In
`NSRulerView.h`

setClientView:

Sets the receiver's client view to *aView*, without retaining it, and removes its ruler markers, after informing the prior client of the change using `rulerView:willSetClientView:` (page 2267).

```
- (void)setClientView:(NSView *)aView
```

Discussion
aView must be either the document view of the NSScrollView that contains the receiver or a subview of the document view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `clientView` (page 2252)

Related Sample Code
Rulers

Declared In
`NSRulerView.h`

setMarkers:

Sets the receiver's ruler markers to *markers*, removing any existing ruler markers and not consulting with the client view about the new markers.

```
- (void)setMarkers:(NSArray *)markers
```

Discussion
markers can be `nil` or empty to remove all ruler markers. Raises an `NSInternalInconsistencyException` if *markers* is not `nil` and the receiver has no client view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addMarker:` (page 2251)
- `removeMarker:` (page 2256)

Related Sample Code
Rulers
Sketch-112

Declared In
`NSRulerView.h`

setMeasurementUnits:

Sets the measurement units used by the ruler to *unitName*.

```
- (void)setMeasurementUnits:(NSString *)unitName
```

Discussion
unitName must have been registered with the NSRulerView class object prior to invoking this method. See the description of the class method `registerUnitWithName:abbreviation:unitToPointsConversionFactor:stepUpCycle:stepDownCycle:` (page 2250) for a list of predefined units.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `measurementUnits` (page 2254)

Related Sample Code
Rulers

Declared In
`NSRulerView.h`

setOrientation:

Sets the orientation of the receiver to *orientation*.

```
- (void)setOrientation:(NSRulerOrientation)orientation
```

Discussion
Possible values for *orientation* are described in "Constants" (page 2267). You should never need to invoke this method directly—it's automatically invoked by the containing NSScrollView.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- orientation` (page 2255)

Declared In
`NSRulerView.h`

setOriginOffset:

Sets the distance to the zero hash mark from the bounds origin of the NSScrollView's document view (not of the receiver's client view), in the document view's coordinate system.

```
- (void)setOriginOffset:(CGFloat)offset
```

Discussion
The default offset is 0.0, meaning that the ruler origin coincides with the bounds origin of the document view.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- originOffset` (page 2256)

Related Sample Code
Rulers

Declared In
`NSRulerView.h`

setReservedThicknessForAccessoryView:

Sets the room available for the receiver's accessory view to *thickness*.

```
- (void)setReservedThicknessForAccessoryView:(CGFloat)thickness
```

Discussion
If the ruler is horizontal, *thickness* is the height of the accessory view; otherwise, it's the width. NSRulerViews by default reserve no space for an accessory view.

An NSRulerView automatically increases the reserved thickness as necessary to that of the accessory view. When the accessory view is thinner than the reserved space, it's centered in that space. If you plan to use several accessory views of different sizes, you should set the reserved thickness beforehand to that of the thickest accessory view, in order to avoid retiling of the NSScrollView.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `reservedThicknessForAccessoryView` (page 2257)
- `setAccessoryView:` (page 2258)
- `setReservedThicknessForMarkers:` (page 2261)

Related Sample Code
Sketch+Accessibility
Sketch-112

Declared In
`NSRulerView.h`

setReservedThicknessForMarkers:

Sets the room available for ruler markers to *thickness*.

```
- (void)setReservedThicknessForMarkers:(CGFloat)thickness
```

Discussion
The default thickness reserved for markers is 15.0 PostScript units for a horizontal ruler and 0.0 PostScript units for a vertical ruler (under the assumption that vertical rulers rarely contain markers). If you don't expect to have any markers on the ruler, you can set the reserved thickness to 0.0.

An NSRulerView automatically increases the reserved thickness as necessary to that of its thickest marker. If you plan to use markers of varying sizes, you should set the reserved thickness beforehand to that of the thickest one in order to avoid retiling of the NSScrollView.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `reservedThicknessForMarkers` (page 2257)
- `setMarkers:` (page 2259)
- `setReservedThicknessForAccessoryView:` (page 2260)
- `thicknessRequiredInRuler` (page 2243) (NSRulerMarker)

Related Sample Code
Sketch-112

Declared In
NSRulerView.h

setRuleThickness:

Sets to *thickness* the thickness of the area where ruler hash marks and labels are drawn.

```
- (void)setRuleThickness:(CGFloat)thickness
```

Discussion
This value is the height of the ruler area for a horizontal ruler or the width of the ruler area for a vertical ruler. Rulers are by default 16.0 PostScript units thick. You should rarely need to change this layout attribute, but subclasses might do so to accommodate custom drawing.

Availability
Available in Mac OS X v10.0 and later.

See Also
- ruleThickness (page 2257)

Declared In
NSRulerView.h

setScrollView:

Sets the NSScrollView that owns the receiver to *scrollView*, without retaining it.

```
- (void)setScrollView:(NSScrollView *)scrollView
```

Discussion
This method is generally invoked only by the ruler's scroll view; you should rarely need to invoke it directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- scrollView (page 2258)
- setHorizontalRulerView: (page 2359) (NSScrollView)
- setVerticalRulerView: (page 2363) (NSScrollView)

Declared In
NSRulerView.h

trackMarker:withMouseEvent:

Tracks the mouse to add *aMarker* based on the initial mouse-down or mouse-dragged event *theEvent*.

```
- (BOOL)trackMarker:(NSRulerMarker *)aMarker withMouseEvent:(NSEvent *)theEvent
```

Discussion
Returns YES if the receiver adds *aMarker*, NO if it doesn't. This method works by sending trackMouse:adding: (page 2243) to *aMarker* with *theEvent* and YES as arguments.

An application typically invokes this method in one of two cases. In the simpler case, the client view can implement `rulerView:handleMouseDown:` (page 2264) to invoke this method when the user presses the mouse button while the cursor is in the NSRulerView's ruler area. This technique is appropriate when it's clear what kind of marker will be added by clicking the ruler area. The second, more general, case involves the application providing a palette of different kinds of markers that can be dragged onto the ruler, from the ruler's accessory view or from some other place. With this technique the palette tracks the cursor until it enters the ruler view, at which time it hands over control to the ruler view by invoking `trackMarker:withMouseEvent:` (page 2262).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addMarker:` (page 2251)
- `setMarkers:` (page 2259)

Related Sample Code
Rulers

Declared In
`NSRulerView.h`

Delegate Methods

rulerView:didAddMarker:

Informs the client that *aRulerView* allowed the user to add *aMarker*.

```
- (void)rulerView:(NSRulerView *)aRulerView didAddMarker:(NSRulerMarker *)aMarker
```

Discussion
The client can take whatever action it needs based on this message, such as adding a new tab stop to the selected paragraph or creating a layout guideline.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `representedObject` (page 2239) (NSRulerMarker)
- `markerLocation` (page 2239) (NSRulerMarker)

Declared In
`NSRulerView.h`

rulerView:didMoveMarker:

Informs the client that *aRulerView* allowed the user to move *aMarker*.

```
- (void)rulerView:(NSRulerView *)aRulerView didMoveMarker:(NSRulerMarker *)aMarker
```

Discussion
The client can take whatever action it needs based on this message, such as updating the location of a tab stop in the selected paragraph, moving a layout guideline, or resizing a graphics element.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `representedObject` (page 2239) (NSRulerMarker)
- `markerLocation` (page 2239) (NSRulerMarker)

Declared In
`NSRulerView.h`

rulerView:didRemoveMarker:

Informs the client that *aRulerView* allowed the user to remove *aMarker*.

```
- (void)rulerView:(NSRulerView *)aRulerView didRemoveMarker:(NSRulerMarker *)aMarker
```

Discussion
The client can take whatever action it needs based on this message, such as deleting a tab stop from the paragraph style or removing a layout guideline.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `representedObject` (page 2239) (NSRulerMarker)

Declared In
`NSRulerView.h`

rulerView:handleMouseDown:

Informs the client that the user has pressed the mouse button while the cursor is in the ruler area of *aRulerView*.

```
- (void)rulerView:(NSRulerView *)aRulerView handleMouseDown:(NSEvent *)theEvent
```

Discussion
theEvent is the mouse-down event that triggered the message. The client view can implement this method to perform an action such as adding a new marker using `trackMarker:withMouseEvent:` (page 2262) or adding layout guidelines.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSRulerView.h`

rulerView:shouldAddMarker:

Requests permission for *aRulerView* to add *aMarker*, an NSRulerMarker being dragged onto the ruler by the user.

```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldAddMarker:(NSRulerMarker *)aMarker
```

Discussion
If the client returns `YES` the ruler view accepts the new marker and begins tracking its movement; if the client returns `NO` the ruler view refuses the new marker.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- rulerView:willAddMarker:atLocation:` (page 2266)

Declared In
`NSRulerView.h`

rulerView:shouldMoveMarker:

Requests permission for *aRulerView* to move *aMarker*.

```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldMoveMarker:(NSRulerMarker *)aMarker
```

Discussion
If the client returns `YES` the ruler view allows the user to move the marker; if the client returns `NO` the marker doesn't move.

The user's ability to move a marker is typically set on the marker itself, using NSRulerMarker's `setMovable:` (page 2242) method. You should use this client view method only when the marker's movability can vary depending on a variable condition (for example, if graphic items can be locked down to prevent them from being inadvertently moved).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- rulerView:willMoveMarker:toLocation:` (page 2266)

Declared In
`NSRulerView.h`

rulerView:shouldRemoveMarker:

Requests permission for *aRulerView* to remove *aMarker*.

```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldRemoveMarker:(NSRulerMarker
    *)aMarker
```

Discussion
If the client returns `YES` the ruler view allows the user to remove the marker; if the client returns `NO` the marker is kept pinned to the ruler's baseline. This message is sent as many times as needed while the user drags the marker.

The user's ability to remove a marker is typically set on the marker itself, using NSRulerMarker's `setRemovable:` (page 2242) method. You should use this client view method only when the marker's removability can vary while the user drags it (for example, if the user must press the Shift key to remove a marker).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSRulerView.h`

rulerView:willAddMarker:atLocation:

Informs the client that *aRulerView* will add the new NSRulerMarker, *aMarker*.

```
- (CGFloat)rulerView:(NSRulerView *)aRulerView willAddMarker:(NSRulerMarker *)aMarker
    atLocation:(CGFloat)location
```

Discussion
location is the marker's tentative new location, expressed in the client view's coordinate system. The value returned by the client view is actually used; the client can simply return *location* unchanged or adjust it as needed. For example, it may snap the location to a grid. This message is sent repeatedly to the client as the user drags the marker.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- rulerView:willMoveMarker:toLocation:` (page 2266)

Declared In
`NSRulerView.h`

rulerView:willMoveMarker:toLocation:

Informs the client that *aRulerView* will move *aMarker*, an NSRulerMarker already on the ruler view.

```
- (CGFloat)rulerView:(NSRulerView *)aRulerView willMoveMarker:(NSRulerMarker
    *)aMarker toLocation:(CGFloat)location
```

Discussion
location is the marker's tentative new location, expressed in the client view's coordinate system. The value returned by the client view is actually used; the client can simply return *location* unchanged or adjust it as needed. For example, it may snap the location to a grid. This message is sent repeatedly to the client as the user drags the marker.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rulerView:willAddMarker:atLocation:` (page 2266)

Declared In
`NSRulerView.h`

rulerView:willSetClientView:

Informs the client view that *aRulerView* is about to be appropriated by *newClient*.

```
- (void)rulerView:(NSRulerView *)aRulerView willSetClientView:(NSView *)newClient
```

Discussion
The client view can use this opportunity to clear any cached information related to the ruler.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSRulerView.h`

Constants

NSRulerOrientation

These constants are defined to specify a ruler's orientation and are used by `orientation` (page 2255) and `setOrientation:` (page 2260).

```
typedef enum {
    NSHorizontalRuler,
    NSVerticalRuler
} NSRulerOrientation;
```

Constants

`NSHorizontalRuler`
Ruler is oriented horizontally.
Available in Mac OS X v10.0 and later.
Declared in `NSRulerView.h`.

`NSVerticalRuler`
Ruler is oriented vertically.
Available in Mac OS X v10.0 and later.
Declared in `NSRulerView.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSRulerView.h`

NSRunningApplication Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSRunningApplication.h
Related sample code	AppList UIElementInspector

Overview

`NSRunningApplication` is a class to manipulate and provide information for a single instance of an application. Only user applications are tracked; this does not provide information about every process on the system.

Some properties of an application are fixed, such as the bundle identifier. Other properties may vary over time, such as whether the app is hidden. Properties that vary can be observed with key-value observing, in which case the description comment for the method notes this capability.

Properties that vary over time are inherently race-prone. For example, a hidden app may unhide itself at any time. To ameliorate this, properties persist until the next turn of the main run loop in a common mode. For example, if you repeatedly poll an unhidden app for its hidden property without allowing the run loop to run, it will continue to return `NO`, even if the app hides, until the next turn of the run loop.

`NSRunningApplication` is thread safe, in that its properties are returned atomically. However, it is still subject to the main run loop policy described above. If you access an instance of NSRunningApplication from a background thread, be aware that its time-varying properties may change from under you as the main run loop runs (or not).

An `NSRunningApplication` instance remains valid after the application exits. However, most properties lose their significance, and some properties may not be available on a terminated application.

To access the list of all running applications, use the `runningApplications` method in `NSWorkspace`.

Tasks

Getting Running Application Instances

+ `runningApplicationWithProcessIdentifier:` (page 2276)
  Returns the running application with the given process identifier, or nil if no application has that pid.

+ `runningApplicationsWithBundleIdentifier:` (page 2276)
  Returns an array of currently running applications with the specified bundle identifier.

+ `currentApplication` (page 2275)
  Returns an `NSRunningApplication` representing this application.

Activating Applications

`active` (page 2271) *property*
  Indicates whether the application is currently frontmost. (read-only)

- `activateWithOptions:` (page 2277)
  Attempts to activate the application using the specified options.

`activationPolicy` (page 2271) *property*
  Indicates the activation policy of the application. (read-only)

Hiding and Unhiding Applications

- `hide` (page 2278)
  Attempts to hide or the application.

- `unhide` (page 2279)
  Attempts to unhide or the application.

`hidden` (page 2273) *property*
  Indicates whether the application is currently hidden. (read-only)

Application Information

`localizedName` (page 2274) *property*
  Indicates the localized name of the application. (read-only)

`icon` (page 2274) *property*
  Returns the icon for the receiver's application. (read-only)

`bundleIdentifier` (page 2272) *property*
  Indicates the `CFBundleIdentifier` of the application. (read-only)

`bundleURL` (page 2272) *property*
  Indicates the URL to the application's bundle. (read-only)

`executableArchitecture` (page 2272) *property*
  Indicates the executing processor architecture for the application. (read-only)

`executableURL` (page 2273) *property*
Indicates the URL to the application's executable. (read-only)

`launchDate` (page 2274) *property*
Indicates the date when the application was launched. (read-only)

`finishedLaunching` (page 2273) *property*
Indicates whether the receiver's process has finished launching, (read-only)

`processIdentifier` (page 2275) *property*
Indicates the process identifier (pid) of the application. (read-only)

Terminating Applications

- `forceTerminate` (page 2277)
Attempts to force the receiver to quit.

- `terminate` (page 2278)
Attempts to quit the receiver normally.

`terminated` (page 2275) *property*
Indicates that the receiver's application has terminated. (read-only)

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

activationPolicy

Indicates the activation policy of the application. (read-only)

```
@property(readonly) NSApplicationActivationPolicy activationPolicy
```

Discussion
The value returned by this property is usually fixed, but it may change through a call to `activateWithOptions:` (page 2277).

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSRunningApplication.h`

active

Indicates whether the application is currently frontmost. (read-only)

```
@property(readonly, getter=isActive) BOOL active
```

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSRunningApplication.h`

bundleIdentifier

Indicates the `CFBundleIdentifier` of the application. (read-only)

```
@property(readonly) NSString *bundleIdentifier
```

Discussion
The value of this property will be `nil` if the application does not have an Info.plist.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSRunningApplication.h`

bundleURL

Indicates the URL to the application's bundle. (read-only)

```
@property(readonly) NSURL *bundleURL
```

Discussion
The value of this property is `nil` of the application does not have a bundle structure.

Availability
Available in Mac OS X v10.6 and later.

See Also
`@property executableURL` (page 2273)

Declared In
`NSRunningApplication.h`

executableArchitecture

Indicates the executing processor architecture for the application. (read-only)

```
@property(readonly) NSInteger executableArchitecture
```

Discussion
The returned value will be one of the constants in `Mach_O_Architecture` in *NSBundle Class Reference*.

Availability
Available in Mac OS X v10.6 and later.

See Also
`@property executableURL` (page 2273)

Declared In
`NSRunningApplication.h`

executableURL

Indicates the URL to the application's executable. (read-only)

```
@property(readonly) NSURL *executableURL
```

Availability
Available in Mac OS X v10.6 and later.

See Also
`@property executableArchitecture` (page 2272)

Declared In
`NSRunningApplication.h`

finishedLaunching

Indicates whether the receiver's process has finished launching, (read-only)

```
@property(readonly, getter=isFinishedLaunching) BOOL finishedLaunching
```

Discussion
The value of this property corresponds to the running application having received an `NSApplicationDidFinishLaunchingNotification` (page 194) notification internally. Some applications do not post this notification (applications that do not rely on `NSApplication`) and so are never reported as finished launching.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSRunningApplication.h`

hidden

Indicates whether the application is currently hidden. (read-only)

```
@property(readonly, getter=isHidden) BOOL hidden
```

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `hide` (page 2278)
- `unhide` (page 2279)

Declared In
`NSRunningApplication.h`

icon

Returns the icon for the receiver's application. (read-only)

```
@property(readonly) NSImage *icon
```

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSRunningApplication.h`

launchDate

Indicates the date when the application was launched. (read-only)

```
@property(readonly) NSDate *launchDate
```

Discussion
This property is not available for all applications. Specifically, it is not available for applications that were launched not launched by LaunchServices.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSRunningApplication.h`

localizedName

Indicates the localized name of the application. (read-only)

```
@property(readonly) NSString *localizedName
```

Discussion
The value of this property is dependent on the current localization of the application and is suitable for presentation to the user.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSRunningApplication.h

processIdentifier

Indicates the process identifier (pid) of the application. (read-only)

```
@property(readonly) pid_t processIdentifier
```

Discussion
Not all applications have a pid. Applications without a pid return a value of -1.

Do not rely on this for comparing processes, instead compare NSRunningApplication instances using `isEqual:`.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSRunningApplication.h

terminated

Indicates that the receiver's application has terminated. (read-only)

```
@property(readonly, getter=isTerminated) BOOL terminated
```

Discussion
The value of terminated is `YES` if the receiver's application has terminated, otherwise `NO`.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- terminate` (page 2278)

Declared In
NSRunningApplication.h

Class Methods

currentApplication

Returns an `NSRunningApplication` representing this application.

```
+ (NSRunningApplication *)currentApplication
```

Return Value
An `NSRunningApplication` instance for the current application.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSRunningApplication.h`

runningApplicationsWithBundleIdentifier:

Returns an array of currently running applications with the specified bundle identifier.

```
+ (NSArray *)runningApplicationsWithBundleIdentifier:(NSString *)bundleIdentifier
```

Parameters
bundleIdentifier
The bundle identifier.

Return Value
An array of `NSRunningApplications`, or an empty array if no applications match the bundle identifier.

Availability
Available in Mac OS X v10.6 and later.

See Also
`+ runningApplicationWithProcessIdentifier:` (page 2276)

Declared In
`NSRunningApplication.h`

runningApplicationWithProcessIdentifier:

Returns the running application with the given process identifier, or nil if no application has that pid.

```
+ (NSRunningApplication *)runningApplicationWithProcessIdentifier:(pid_t)pid
```

Parameters
pid
The process identifier.

Return Value
An instance of `NSRunningApplication` for the specified *pid*, or nil if the application has no process identifier.

Discussion
Applications that do not have `PIDs` cannot be returned from this method.

Availability
Available in Mac OS X v10.6 and later.

See Also
`+ runningApplicationsWithBundleIdentifier:` (page 2276)

Related Sample Code
UIElementInspector

Declared In
`NSRunningApplication.h`

Instance Methods

activateWithOptions:

Attempts to activate the application using the specified options.

```
- (BOOL)activateWithOptions:(NSApplicationActivationOptions)options
```

Parameters

options

The options to use when activating the application. See "NSApplicationActivationOptions" (page 2279) for the possible values.

Return Value
`YES` if the application was activated successfully, otherwise `NO`.

Discussion
This method will return `NO` if the application has quit, or is not a type of application than can be activated.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSRunningApplication.h`

forceTerminate

Attempts to force the receiver to quit.

```
- (BOOL)forceTerminate
```

Return Value
Returns `YES` if the application successfully terminated, otherwise `NO`.

Discussion
This method will return `NO` if the application is no longer running when the `forceTerminate` message is sent to the receiver.

This method may return before the receiver exits; you should observe the terminated property to determine when the application terminates.

Availability
Available in Mac OS X v10.6 and later.

See Also
`@property terminated` (page 2275)

- terminate (page 2278)

Declared In
NSRunningApplication.h

hide

Attempts to hide or the application.

- (BOOL)hide

Return Value
YES if the application was successfully hidden, otherwise NO.

Discussion
The property of this value will be NO if the application has already quit, or if of a type that is unable to be hidden.

Availability
Available in Mac OS X v10.6 and later.

See Also
- unhide (page 2279)
@property hidden (page 2273)

Related Sample Code
AppList

Declared In
NSRunningApplication.h

terminate

Attempts to quit the receiver normally.

- (BOOL)terminate

Return Value
Returns YES if the application successfully terminated, otherwise NO.

Discussion
This method will return NO if the application is no longer running when the terminate message is sent to the receiver.

This method may return before the receiver exits; you should observe the terminated property to determine when the application terminates.

Availability
Available in Mac OS X v10.6 and later.

See Also
@property terminated (page 2275)

Related Sample Code
AppList

Declared In
`NSRunningApplication.h`

unhide

Attempts to unhide or the application.

```
- (BOOL)unhide
```

Return Value
`YES` if the application was successfully shown, otherwise `NO`.

Discussion
The property of this value will be `NO` if the application has already quit, or if of a type that is unable to be hidden.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- hide` (page 2278)
`@property hidden` (page 2273)

Related Sample Code
AppList

Declared In
`NSRunningApplication.h`

Constants

NSApplicationActivationOptions

The following flags are for `activateWithOptions:` (page 2277).

```
enum {
    NSApplicationActivateAllWindows = 1 << 0,
    NSApplicationActivateIgnoringOtherApps = 1 << 1
};
typedef NSUInteger NSApplicationActivationOptions;
```

Constants
`NSApplicationActivateAllWindows`
By default, activation brings only the main and key windows forward. If you specify NSApplicationActivateAllWindows, all of the application's windows are brought forward.

Available in Mac OS X v10.6 and later.

Declared in `NSRunningApplication.h`.

`NSApplicationActivateIgnoringOtherApps`
: By default, activation deactivates the calling app (assuming it was active), and then the new app is activated only if there is no currently active application. This prevents the new app from stealing focus from the user, if the app is slow to activate and the user has switched to a different app in the interim. However, if you specify NSApplicationActivateIgnoringOtherApps, the application is activated regardless of the currently active app, potentially stealing focus from the user. You should **rarely pass this flag** because stealing key focus produces a very bad user experience.

  Available in Mac OS X v10.6 and later.

  Declared in `NSRunningApplication.h`.

NSApplicationActivationPolicy

The following activation policies control whether and how an application may be activated. They are used by `activationPolicy` (page 2271).

```
enum {
    NSApplicationActivationPolicyRegular,
    NSApplicationActivationPolicyAccessory,
    NSApplicationActivationPolicyProhibited
};
typedef NSInteger NSApplicationActivationPolicy;
```

Constants

`NSApplicationActivationPolicyRegular`
: The application is an ordinary app that appears in the Dock and may have a user interface. This is the default for bundled apps, unless overridden in the `Info.plist`.

  Available in Mac OS X v10.6 and later.

  Declared in `NSRunningApplication.h`.

`NSApplicationActivationPolicyAccessory`
: The application does not appear in the Dock and does not have a menu bar, but it may be activated programmatically or by clicking on one of its windows. This corresponds to value of the `LSUIElement` key in the application's `Info.plist` being `1`.

  Available in Mac OS X v10.6 and later.

  Declared in `NSRunningApplication.h`.

`NSApplicationActivationPolicyProhibited`
: The application does not appear in the Dock and may not create windows or be activated. This corresponds to the value of the `LSBackgroundOnly` key in the application's `Info.plist` being `1`. This is also the default for unbundled executables that do not have `Info.plists`.

  Available in Mac OS X v10.6 and later.

  Declared in `NSRunningApplication.h`.

NSSavePanel Class Reference

Inherits from	NSPanel : NSWindow : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSWindow) NSAnimatablePropertyContainer (NSWindow) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSSavePanel.h
Companion guides	Application File Management Sheet Programming Topics
Related sample code	ImageKitDemo QTAudioContextInsert QTAudioExtractionPanel Quartz Composer WWDC 2005 TextEdit WhackedTV

Overview

An `NSSavePanel` object creates and manages a Save panel and allows you to run the panel in a modal loop. The Save panel provides a simple way for a user to specify a file to use when saving a document or other data. It can restrict the user to files of a certain type, as specified by an extension.

An `NSSavePanel` object manages a panel that allows users to specify the directory and name under which a file is saved. It supports browsing of the file system, and it accommodates custom accessory views.

An `NSSavePanel` object may have a delegate. The methods that delegates of `NSSavePanel` may implement are specified by the `NSOpenSavePanelDelegate` protocol.

Tasks

Creating Panels

+ `savePanel` (page 2285)
  Returns a Save panel that has been initialized with default values.

Configuring Panels

- `accessoryView` (page 2285)
  Returns the custom accessory view for the current application.
- `setAccessoryView:` (page 2296)
  Customizes the panel for the application by adding a custom view to the panel.
- `title` (page 2306)
  Returns the title of the panel.
- `setTitle:` (page 2305)
  Sets the title of the panel.
- `prompt` (page 2294)
  Returns the prompt of the default button.
- `setPrompt:` (page 2303)
  Sets the prompt of the default button.
- `nameFieldLabel` (page 2293)
  Returns the string displayed in front of the filename text field.
- `setNameFieldLabel:` (page 2302)
  Sets the text displayed in front of the text field.
- `message` (page 2292)
  Returns the message displayed in the save panel.
- `setMessage:` (page 2301)
  Sets the message text displayed in the panel.
- `canCreateDirectories` (page 2289)
  Returns a Boolean value that indicates whether the panel allows the user to create directories.
- `setCanCreateDirectories:` (page 2298)
  Sets whether the panel allows the user to create directories.
- `showsHiddenFiles` (page 2306)
  Returns whether the panel displays files that are normally hidden from the user.
- `setShowsHiddenFiles:` (page 2304)
  Specifies whether the panel displays files that are normally hidden from the user.
- `delegate` (page 2290)
  Returns the panel's delegate.
- `setDelegate:` (page 2299)
  Sets an object as the panel's delegate, after verifying which delegate methods are implemented.

Configuring Panel Content

- `isExtensionHidden` (page 2292)
  Returns a Boolean value that indicates whether the extension-hiding checkbox is visible and checked.
- `setExtensionHidden:` (page 2301)
  Sets the value of the extension-hiding checkbox.
- `canSelectHiddenExtension` (page 2290)
  Returns a Boolean value that indicates whether the panel allows the user to hide or show extensions.
- `setCanSelectHiddenExtension:` (page 2299)
  Sets whether the panel allows the user to hide or show extensions.
- `allowedFileTypes` (page 2286)
  Returns an array of the allowed file types.
- `setAllowedFileTypes:` (page 2297)
  Specifies the allowed file types for the panel.
- `allowsOtherFileTypes` (page 2286)
  Returns a Boolean value that indicates whether the panel allows the user to save files with an extension that's not in the list of allowed types.
- `setAllowsOtherFileTypes:` (page 2298)
  Sets whether the panel allows the user to save files with an extension that's not in the list of allowed types.
- `treatsFilePackagesAsDirectories` (page 2306)
  Returns a Boolean value that indicates whether the panel displays file packages as directories.
- `setTreatsFilePackagesAsDirectories:` (page 2305)
  Sets the panel's behavior for displaying file packages (for example, `MyApp.app`) to the user.
- `requiredFileType` (page 2294) Deprecated in Mac OS X v10.6
  Returns the required file type (if any). (Deprecated. Use `allowedFileTypes` (page 2286) instead.)
- `setRequiredFileType:` (page 2303) Deprecated in Mac OS X v10.6
  Specifies the type, an extension to be appended to any selected files that don't already have that extension; "nib" and "rtf" are examples. (Deprecated. Use `setAllowedFileTypes:` (page 2297) instead.)

Running Panels

- `beginSheetModalForWindow:completionHandler:` (page 2288)
  Presents the panel as a sheet modal to the specified window.
- `beginWithCompletionHandler:` (page 2288)
  Presents the panel as a modeless window.
- `runModal` (page 2295)
  Displays the panel and begins its event loop with the current working (or last selected) directory as the default starting point.
- `validateVisibleColumns` (page 2307)
  Validates and possibly reloads the browser columns visible in the panel by invoking the delegate method `panel:shouldShowFilename:` (page 2309).

- `beginSheetForDirectory:file:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 2287) Deprecated in Mac OS X v10.6
  Presents a Save panel as a sheet with a specified path and, optionally, a specified file in that path. (Deprecated. Use `beginSheetModalForWindow:completionHandler:` (page 2288) instead.)
- `runModalForDirectory:file:` (page 2295) Deprecated in Mac OS X v10.6
  Initializes the panel to the directory and file specified, if any, then displays it and begins its modal event loop. (Deprecated. Use `runModal` (page 2295) instead.)
- `panel:compareFilename:with:caseSensitive:` (page 2307) *delegate method* Deprecated in Mac OS X v10.6
  Controls the ordering of files presented by the `NSSavePanel` object specified. (Deprecated. There is no replacement.)
- `panel:directoryDidChange:` (page 2308) *delegate method* Deprecated in Mac OS X v10.6
  Tells the delegate that the user has changed the selected directory in the `NSSavePanel` object specified. (Deprecated. Use `panel:didChangeToDirectoryURL:` (page 3736) (`NSOpenSavePanelDelegate`) instead.)
- `panel:isValidFilename:` (page 2309) *delegate method* Deprecated in Mac OS X v10.6
  Gives the delegate the opportunity to validate selected items. (Deprecated. Use `panel:validateURL:error:` (page 3737) (`NSOpenSavePanelDelegate`) instead. If both methods are implemented, the URL version will be called.)
- `panel:shouldShowFilename:` (page 2309) *delegate method* Deprecated in Mac OS X v10.6
  Gives the delegate the opportunity to filter items that it doesn't want the user to choose. (Deprecated. Use `panel:shouldEnableURL:` (page 3736) (`NSOpenSavePanelDelegate`).)

Accessing User Selection

- `directoryURL` (page 2291)
  Returns the directory shown in the panel as a URL.
- `setDirectoryURL:` (page 2300)
  Sets the directory shown in the panel to the directory with the specified URL.
- `URL` (page 2307)
  Returns the absolute pathname of the file currently shown in the panel as a URL.
- `isExpanded` (page 2292)
  Returns a Boolean value that indicates whether the panel is expanded.
- `nameFieldStringValue` (page 2293)
  Returns the user-editable filename currently shown in the name field.
- `setNameFieldStringValue:` (page 2302)
  Sets the filename in the name field to the specified value.
- `selectText:` (page 2296) Deprecated in Mac OS X v10.3
  This method has been deprecated. (Deprecated. There is no replacement.)
- `directory` (page 2290) Deprecated in Mac OS X v10.6
  Returns the absolute pathname of the directory currently shown in the panel. (Deprecated. Use `directoryURL` (page 2291) instead.)
- `filename` (page 2291) Deprecated in Mac OS X v10.6
  Returns the absolute pathname of the file currently shown in the panel. (Deprecated. Use `URL` (page 2307) instead.)

- `setDirectory:` (page 2300) Deprecated in Mac OS X v10.6
  Sets the current pathname in the panel's browser. (Deprecated. Use `setDirectoryURL:` (page 2300) instead.)

Handling Actions

- `ok:` (page 2294)
  This action method is invoked when the user clicks the panel's OK button.
- `cancel:` (page 2289)
  This action method is invoked when the user clicks the panel's Cancel button.

Class Methods

savePanel

Returns a Save panel that has been initialized with default values.

```
+ (NSSavePanel *)savePanel
```

Return Value
The initialized Save panel.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CIVideoDemoGL
ImageKitDemo
QTAudioExtractionPanel
Quartz Composer WWDC 2005 TextEdit
WhackedTV

Declared In
`NSSavePanel.h`

Instance Methods

accessoryView

Returns the custom accessory view for the current application.

```
- (NSView *)accessoryView
```

Return Value
The custom accessory view for the current application.

Discussion
In order to free up unused memory after closing the panel, the accessory view is released after the panel is closed. If you rely on the panel to hold onto the accessory view until the next time you use it, the accessory view may be deallocated unexpectedly. If you retain the accessory view in your own code, though, this deallocation should not be a problem.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAccessoryView:` (page 2296)

Declared In
`NSSavePanel.h`

allowedFileTypes

Returns an array of the allowed file types.

```
- (NSArray *)allowedFileTypes
```

Return Value
An array of the allowed file types.

Discussion
If the user specifies a file whose type is in the array of allowed types, the user is not presented with another dialog (see `allowsOtherFileTypes` (page 2286) for details about this dialog) when trying to save. Examples of common file types are "rtf", "tiff", and "ps". File type strings encoding HFS file types are not valid values for this attribute. A `nil` return value, which is the default, indicates that the user can save to any ASCII file.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setAllowedFileTypes:` (page 2297)

Declared In
`NSSavePanel.h`

allowsOtherFileTypes

Returns a Boolean value that indicates whether the panel allows the user to save files with an extension that's not in the list of allowed types.

```
- (BOOL)allowsOtherFileTypes
```

Return Value
`YES` if the panel allows the user to save files with an extension that's not in the list of allowed types; otherwise, `NO`.

Discussion
If the user tries to save a filename with a recognized extension that's not in the list of allowed types they are presented with a dialog. If this method returns `YES`, then the dialog presents the option of using the extension the user specified.

The default setting is `NO`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setAllowsOtherFileTypes:` (page 2298)
- `allowedFileTypes` (page 2286)

Declared In
`NSSavePanel.h`

beginSheetForDirectory:file:modalForWindow:modalDelegate:didEndSelector:contextInfo:

Presents a Save panel as a sheet with a specified path and, optionally, a specified file in that path. (Deprecated in Mac OS X v10.6. Use `beginSheetModalForWindow:completionHandler:` (page 2288) instead.)

```
- (void)beginSheetForDirectory:(NSString *)path file:(NSString *)name
    modalForWindow:(NSWindow *)docWindow modalDelegate:(id)modalDelegate
    didEndSelector:(SEL)didEndSelector contextInfo:(void *)contextInfo
```

Parameters

path

Directory whose files the panel displays. When `nil`, the directory is the same directory used in the previous invocation of the panel; this is probably the best choice for most situations.

name

Specifies a particular file in *path* that is selected when the Save panel is presented to a user. When `nil`, no file is initially selected.

docWindow

If not `nil`, the Save panel slides down as a sheet running as a document modal window in *docWindow*. If `nil`, the behavior defaults to a standalone modal window.

modalDelegate

This is not the same as a delegate assigned to the panel. This delegate is temporary and the relationship only lasts until the panel is dismissed. The `NSSavePanel` object does not retain the modal delegate.

didEndSelector

Message sent to *modalDelegate* after the modal session has ended, but before dismissing the Save panel. *didEndSelector* may dismiss the Save panel itself; otherwise, it is dismissed on return from the method. The corresponding method should have the following signature:

```
- (void)savePanelDidEnd:(NSSavePanel *)sheet returnCode:(int)returnCode
contextInfo:(void *)contextInfo;
```

The value passed as *returnCode* is either `NSCancelButton` or `NSOKButton`.

contextInfo

Context information passed to *modalDelegate* in the *didEndSelector* message.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
`- beginSheetModalForWindow:completionHandler:` (page 2288)

Related Sample Code
Cropped Image
ImageKitDemo
QTAudioExtractionPanel
Quartz Composer WWDC 2005 TextEdit
RGB Image

Declared In
`NSSavePanel.h`

beginSheetModalForWindow:completionHandler:

Presents the panel as a sheet modal to the specified window.

```
- (void)beginSheetModalForWindow:(NSWindow *)window
    completionHandler:(void (^)(NSInteger result))handler
```

Parameters

window
The window in which the panel will be presented.

handler
The block called after the user has closed the panel. The argument passed in will be `NSFileHandlingPanelOKButton` if the user chose the OK button or `NSFileHandlingPanelCancelButton` if the user chose the Cancel button.

Discussion
Any properties of the panel you wish to set should be set before calling this method.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
Denoise
DispatchFractal
From A View to A Movie
From A View to A Picture
TextSizingExample

Declared In
`NSSavePanel.h`

beginWithCompletionHandler:

Presents the panel as a modeless window.

```
- (void)beginWithCompletionHandler:(void (^)(NSInteger result))handler
```

Parameters

handler

The block called after the user has closed the panel. The argument passed in will be `NSFileHandlingPanelOKButton` if the user chose the OK button or `NSFileHandlingPanelCancelButton` if the user chose the Cancel button.

Discussion

Any properties of the panel you wish to set should be set before calling this method.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSSavePanel.h`

cancel:

This action method is invoked when the user clicks the panel's Cancel button.

```
- (IBAction)cancel:(id)sender
```

Parameters

sender

The `NSSavePanel` object whose Cancel button was clicked.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- ok:` (page 2294)

Declared In

`NSSavePanel.h`

canCreateDirectories

Returns a Boolean value that indicates whether the panel allows the user to create directories.

```
- (BOOL)canCreateDirectories
```

Return Value

`YES` when the panel allows the user to create directories; otherwise, `NO`.

Discussion

The default value is `YES`.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- setCanCreateDirectories:` (page 2298)

Declared In
NSSavePanel.h

canSelectHiddenExtension

Returns a Boolean value that indicates whether the panel allows the user to hide or show extensions.

```
- (BOOL)canSelectHiddenExtension
```

Return Value
YES when the panel allows the user to hide or show extensions; otherwise, NO.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setCanSelectHiddenExtension: (page 2299)

Declared In
NSSavePanel.h

delegate

Returns the panel's delegate.

```
- (id <NSOpenSavePanelDelegate>)delegate
```

Return Value
The panel's delegate.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setDelegate: (page 2299)

Declared In
NSSavePanel.h

directory

Returns the absolute pathname of the directory currently shown in the panel. (Deprecated in Mac OS X v10.6. Use directoryURL (page 2291) instead.)

```
- (NSString *)directory
```

Return Value
The absolute pathname of the directory currently shown in the panel.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- directoryURL (page 2291)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel

Declared In
NSSavePanel.h

directoryURL

Returns the directory shown in the panel as a URL.

```
- (NSURL *)directoryURL
```

Return Value
The directory's URL.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setDirectoryURL: (page 2300)

Related Sample Code
From A View to A Movie
From A View to A Picture

Declared In
NSSavePanel.h

filename

Returns the absolute pathname of the file currently shown in the panel. (Deprecated in Mac OS X v10.6. Use URL (page 2307) instead.)

```
- (NSString *)filename
```

Return Value
The absolute pathname of the file.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- URL (page 2307)

Related Sample Code
CIVideoDemoGL
ImageKitDemo

QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit
WhackedTV

Declared In
`NSSavePanel.h`

isExpanded

Returns a Boolean value that indicates whether the panel is expanded.

```
- (BOOL)isExpanded
```

Return Value
`YES` if the panel is expanded; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSavePanel.h`

isExtensionHidden

Returns a Boolean value that indicates whether the extension-hiding checkbox is visible and checked.

```
- (BOOL)isExtensionHidden
```

Return Value
`YES` when the extension-hiding checkbox is visible and checked; otherwise, `NO`.

Availability
Available in Mac OS X v10.1 and later.

See Also
`- setCanSelectHiddenExtension:` (page 2299)
`- setExtensionHidden:` (page 2301)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSSavePanel.h`

message

Returns the message displayed in the save panel.

```
- (NSString *)message
```

Return Value
The message displayed in the save panel.

Discussion
This prompt appears on all `NSSavePanel` objects (or all `NSOpenPanel` objects if the receiver of this message is an `NSOpenPanel` instance) in your application. The default message text is an empty string.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setMessage:` (page 2301)

Declared In
`NSSavePanel.h`

nameFieldLabel

Returns the string displayed in front of the filename text field.

```
- (NSString *)nameFieldLabel
```

Return Value
The string displayed in front of the filename text field.

Discussion
By default the label is “Save As:”.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setNameFieldLabel:` (page 2302)

Declared In
`NSSavePanel.h`

nameFieldStringValue

Returns the user-editable filename currently shown in the name field.

```
- (NSString *)nameFieldStringValue
```

Return Value
The filename currently selected.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setNameFieldStringValue:` (page 2302)

Declared In
`NSSavePanel.h`

ok:

This action method is invoked when the user clicks the panel's OK button.

```
- (IBAction)ok:(id)sender
```

Parameters

sender

The `NSSavePanel` object whose OK button was clicked.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- cancel:` (page 2289)

Declared In
`NSSavePanel.h`

prompt

Returns the prompt of the default button.

```
- (NSString *)prompt
```

Return Value
The prompt of the default button.

Discussion
This prompt appears on all `NSSavePanel` objects (or all `NSOpenPanel` objects if the receiver of this message is an `NSOpenPanel` instance) in your application. By default, the text in the default button is "Open" for an open panel and "Save" for a Save panel.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setPrompt:` (page 2303)

Declared In
`NSSavePanel.h`

requiredFileType

Returns the required file type (if any). (Deprecated in Mac OS X v10.6. Use `allowedFileTypes` (page 2286) instead.)

```
- (NSString *)requiredFileType
```

Return Value
The required file type (if any).

Discussion
A file specified in the Save panel is saved with the designated filename and this file type as an extension. Examples of common file types are "rtf", "tiff", and "ps". File type strings encoding HFS file types are not valid values for this attribute. An `nil` return value indicates that the user can save to any ASCII file.

This method is equivalent to calling `allowedFileTypes` (page 2286) and returning the first element of the list of allowed types, or `nil` if there are none.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
`- allowedFileTypes` (page 2286)

Declared In
`NSSavePanel.h`

runModal

Displays the panel and begins its event loop with the current working (or last selected) directory as the default starting point.

```
- (NSInteger)runModal
```

Return Value
`NSFileHandlingPanelOKButton` (if the user clicks the OK button) or `NSFileHandlingPanelCancelButton` (if the user clicks the Cancel button).

Discussion
This method invokes `NSApplication`'s `runModalForWindow:` (page 163) method with `self` as the argument.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- runModalForWindow:` (page 163) (`NSApplication`)

Related Sample Code
AudioDataOutputToAudioUnit
QTKitProgressTester
ThreadsExportMovie

Declared In
`NSSavePanel.h`

runModalForDirectory:file:

Initializes the panel to the directory and file specified, if any, then displays it and begins its modal event loop. (Deprecated in Mac OS X v10.6. Use `runModal` (page 2295) instead.)

```
- (NSInteger)runModalForDirectory:(NSString *)path file:(NSString *)filename
```

Parameters

path

Directory whose files the panel displays. When `nil`, the directory is the same directory used in the previous invocation of the panel; this is probably the best choice for most situations.

filename

Specifies a particular file in *path* that is selected when the Save panel is presented to a user. When `nil`, no file is initially selected.

Return Value

`NSFileHandlingPanelOKButton` (if the user clicks the OK button) or `NSFileHandlingPanelCancelButton` (if the user clicks the Cancel button).

Discussion

This method invokes `NSApplication`'s `runModalForWindow:` (page 163) method with `self` as the argument.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- `runModal` (page 2295)
- `runModalForWindow:` (page 163) (`NSApplication`)

Related Sample Code

AlbumToSlideshow

CIFilterGeneratorTest

CoreRecipes

Declared In

`NSSavePanel.h`

selectText:

This method has been deprecated. (Deprecated in Mac OS X v10.3. There is no replacement.)

```
- (IBAction)selectText:(id)sender
```

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Declared In

`NSSavePanel.h`

setAccessoryView:

Customizes the panel for the application by adding a custom view to the panel.

```
- (void)setAccessoryView:(NSView *)aView
```

Parameters

aView

View to set as the panel's accessory view.

Discussion

The custom object that is added appears just above the OK and Cancel buttons at the bottom of the panel. The `NSSavePanel` object automatically resizes itself to accommodate *aView*. You can invoke this method repeatedly to change the accessory view as needed. If *aView* is `nil`, the Save panel removes the current accessory view.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- accessoryView` (page 2285)

Related Sample Code

ImageApp
QTAudioContextInsert
QTAudioExtractionPanel
QTKitPlayer
Quartz Composer WWDC 2005 TextEdit

Declared In

`NSSavePanel.h`

setAllowedFileTypes:

Specifies the allowed file types for the panel.

```
- (void)setAllowedFileTypes:(NSArray *)types
```

Parameters

types

Array to set as the panel's array of allowed file types.

Discussion

The parameter must not be empty. A file type is an extension to be appended to any selected files that don't already have that extension; "nib" and "rtf" are examples. The items in *types* should not include the period that begins the extension. File type strings encoding HFS file types are not valid values. Pass `nil` to allow any file type, which is the default.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- allowedFileTypes` (page 2286)

Related Sample Code

AudioDataOutputToAudioUnit
ImageApp
TimelineToTC

Declared In
NSSavePanel.h

setAllowsOtherFileTypes:

Sets whether the panel allows the user to save files with an extension that's not in the list of allowed types.

```
- (void)setAllowsOtherFileTypes:(BOOL)flag
```

Parameters

flag

If YES, the panel allows the user to save files with an extension that's not in the list of allowed types; if NO, it does not.

Discussion
If the user tries to save a filename with a recognized extension that's not in the list of allowed types they are presented with a dialog. If allowsOtherFileTypes (page 2286) is YES, then the dialog presents the option of using the extension the user specified.

The default setting is NO.

Availability
Available in Mac OS X v10.3 and later.

See Also
- allowsOtherFileTypes (page 2286)
- allowedFileTypes (page 2286)

Related Sample Code
MassiveImage
Quartz 2D Transformer
Quartz Composer WWDC 2005 TextEdit

Declared In
NSSavePanel.h

setCanCreateDirectories:

Sets whether the panel allows the user to create directories.

```
- (void)setCanCreateDirectories:(BOOL)flag
```

Parameters

flag

If YES, the panel allows the user to create directories; if NO, the panel does not.

Availability
Available in Mac OS X v10.3 and later.

See Also
- canCreateDirectories (page 2289)

Related Sample Code
GridCalendar
OpenGLCaptureToMovie
Quartz Composer QCTV
WhackedTV

Declared In
`NSSavePanel.h`

setCanSelectHiddenExtension:

Sets whether the panel allows the user to hide or show extensions.

```
- (void)setCanSelectHiddenExtension:(BOOL)flag
```

Parameters

flag

If `YES`, the panel allows the user to hide or show extensions; if `NO`, it does not.

Discussion
This method must be called before the panel is displayed. If set to `YES`, `isExtensionHidden` and `setExtensionHidden:`, respectively, can be used to get and set the value of the checkbox that hides or shows extensions.

Availability
Available in Mac OS X v10.1 and later.

See Also
- `canSelectHiddenExtension` (page 2290)
- `isExtensionHidden` (page 2292)
- `setExtensionHidden:` (page 2301)

Related Sample Code
AudioDataOutputToAudioUnit
QTRecorder
Quartz 2D Transformer
Quartz Composer WWDC 2005 TextEdit
WhackedTV

Declared In
`NSSavePanel.h`

setDelegate:

Sets an object as the panel's delegate, after verifying which delegate methods are implemented.

```
- (void)setDelegate:(id <NSOpenSavePanelDelegate>)anObject
```

Parameters

anObject

Object to set as the panel's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `delegate` (page 2290)

Related Sample Code
AudioBurn
QTAudioContextInsert
QTMetadataEditor
Quartz Composer WWDC 2005 TextEdit
SBSetFinderComment

Declared In
`NSSavePanel.h`

setDirectory:

Sets the current pathname in the panel's browser. (Deprecated in Mac OS X v10.6. Use `setDirectoryURL:` (page 2300) instead.)

```
- (void)setDirectory:(NSString *)path
```

Parameters

path
String to set as the panel's current pathname.

Discussion
The *path* argument must be an absolute pathname.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `setDirectoryURL:` (page 2300)

Related Sample Code
CustomSave
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSSavePanel.h`

setDirectoryURL:

Sets the directory shown in the panel to the directory with the specified URL.

```
- (void)setDirectoryURL:(NSURL *)url
```

Parameters

url

The URL of the directory to set.

Availability
Available in Mac OS X v10.6 and later.

See Also
- directoryURL (page 2291)

Declared In
NSSavePanel.h

setExtensionHidden:

Sets the value of the extension-hiding checkbox.

```
- (void)setExtensionHidden:(BOOL)flag
```

Parameters

flag

If YES, the extension-hiding checkbox is visible and checked; if NO, it is not.

Discussion
This method should rarely be used because the state is saved on a per-application basis. Use this method to set whether a file's extension should be indicated as being shown.

Availability
Available in Mac OS X v10.1 and later.

See Also
- setCanSelectHiddenExtension: (page 2299)
- isExtensionHidden (page 2292)

Related Sample Code
Fiendishthngs
FunHouse

Declared In
NSSavePanel.h

setMessage:

Sets the message text displayed in the panel.

```
- (void)setMessage:(NSString *)message
```

Parameters

message

String to set as the panel's message.

Discussion
This prompt appears on all `NSSavePanel` objects (or all `NSOpenPanel` objects if the receiver of this message is an `NSOpenPanel` instance) in your application. The default message text is an empty string.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- message` (page 2292)

Related Sample Code
CustomSave
NSOperationSample
Quartz 2D Transformer
SBSetFinderComment
ViewController

Declared In
`NSSavePanel.h`

setNameFieldLabel:

Sets the text displayed in front of the text field.

```
- (void)setNameFieldLabel:(NSString *)label
```

Parameters
label
String to set as the text displayed in front of the panel's text field.

Discussion
By default the label is "Save As:".

Availability
Available in Mac OS X v10.3 and later.

See Also
`- nameFieldLabel` (page 2293)

Related Sample Code
CustomSave

Declared In
`NSSavePanel.h`

setNameFieldStringValue:

Sets the filename in the name field to the specified value.

```
- (void)setNameFieldStringValue:(NSString *)value
```

Parameters

value

The filename to set. The value must not be `nil`.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `nameFieldStringValue` (page 2293)

Declared In
`NSSavePanel.h`

setPrompt:

Sets the prompt of the default button.

```
- (void)setPrompt:(NSString *)prompt
```

Parameters

prompt

String to set as the prompt of the panel's default button.

Discussion
This prompt appears on all `NSSavePanel` objects (or all `NSOpenPanel` objects if the receiver of this message is an `NSOpenPanel` instance) in your application. By default, the text in the default button is "Open" for an Open panel and "Save" for a Save panel.

It is intended that short words or phrases, such as "Open," "Save," "Set," or "Choose," be used on the button. The button is not resized to accommodate long prompts.

Since this method previously affected a title field, any colon at the end of *prompt* is removed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `prompt` (page 2294)

Related Sample Code
AudioBurn
CocoaSpeechSynthesisExample
ContentBurn
DataBurn
ObjectPath

Declared In
`NSSavePanel.h`

setRequiredFileType:

Specifies the type, an extension to be appended to any selected files that don't already have that extension; "nib" and "rtf" are examples. (Deprecated in Mac OS X v10.6. Use `setAllowedFileTypes:` (page 2297) instead.)

```
- (void)setRequiredFileType:(NSString *)type
```

Parameters

type

String to set as the extension to be appended to any selected files that don't already have that extension.

Discussion

The argument *type* should not include the period that begins the extension. Pass `nil` to indicate any type. File type strings encoding HFS file types are not valid values for this attribute. You need to invoke this method each time the Save panel is used for another file type within the application.

This method is equivalent to calling `setAllowedFileTypes:` (page 2297) with an array containing only *type* (unless *type* is nil, and then it's equivalent to calling `setAllowedFileTypes:` with `nil`).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

`- setAllowedFileTypes:` (page 2297)

Related Sample Code

CIVideoDemoGL

ExtractMovieAudioToAIFF

Quartz Composer QCTV

Quartz Composer WWDC 2005 TextEdit

WhackedTV

Declared In

`NSSavePanel.h`

setShowsHiddenFiles:

Specifies whether the panel displays files that are normally hidden from the user.

```
- (void)setShowsHiddenFiles:(BOOL)flag
```

Parameters

flag

If `YES`, the panel displays hidden files; if `NO`, it does not.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- showsHiddenFiles` (page 2306)

Declared In

`NSSavePanel.h`

setTitle:

Sets the title of the panel.

```
- (void)setTitle:(NSString *)title
```

Parameters

title

String to set as the panel's title.

Discussion

By default, "Save" is the title string. If you adapt the `NSSavePanel` object for other uses, its title should reflect the user action that brings it to the screen.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- title` (page 2306)

Related Sample Code

AudioBurn

ContentBurn

MovieAssembler

ObjectPath

Quartz Composer WWDC 2005 TextEdit

Declared In

`NSSavePanel.h`

setTreatsFilePackagesAsDirectories:

Sets the panel's behavior for displaying file packages (for example, `MyApp.app`) to the user.

```
- (void)setTreatsFilePackagesAsDirectories:(BOOL)flag
```

Parameters

flag

If `YES`, the panel will display file packages as directories; if `NO`, it will not.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- treatsFilePackagesAsDirectories` (page 2306)

Related Sample Code

CustomSave

Quartz 2D Transformer

TextSizingExample

Declared In

`NSSavePanel.h`

showsHiddenFiles

Returns whether the panel displays files that are normally hidden from the user.

```
- (BOOL)showsHiddenFiles
```

Return Value
`YES` if the panel displays hidden files; otherwise, `NO`.

Discussion
The default value is `NO`.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setShowsHiddenFiles:` (page 2304)

Declared In
`NSSavePanel.h`

title

Returns the title of the panel.

```
- (NSString *)title
```

Return Value
The title of the panel.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitle:` (page 2305)

Declared In
`NSSavePanel.h`

treatsFilePackagesAsDirectories

Returns a Boolean value that indicates whether the panel displays file packages as directories.

```
- (BOOL)treatsFilePackagesAsDirectories
```

Return Value
`YES` if the panel displays file packages as directories; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTreatsFilePackagesAsDirectories:` (page 2305)

Declared In
NSSavePanel.h

URL

Returns the absolute pathname of the file currently shown in the panel as a URL.

```
- (NSURL *)URL
```

Return Value
The absolute pathname of the file currently shown in the panel as a URL.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaSpeechSynthesisExample
MassiveImage
PDF Annotation Editor
PDF Calendar
Quartz2DBasics

Declared In
NSSavePanel.h

validateVisibleColumns

Validates and possibly reloads the browser columns visible in the panel by invoking the delegate method `panel:shouldShowFilename:` (page 2309).

```
- (void)validateVisibleColumns
```

Discussion
You might use this method if you want the browser to only allow selection of files with certain extensions based on the selection made in an accessory-view pop-up list. When the user changes the selection, you would invoke this method to revalidate the visible columns.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSSavePanel.h

Delegate Methods

panel:compareFilename:with:caseSensitive:

Controls the ordering of files presented by the `NSSavePanel` object specified. (Deprecated in Mac OS X v10.6. There is no replacement.)

```
- (NSComparisonResult)panel:(id)sender compareFilename:(NSString *)fileName1
    with:(NSString *)fileName2 caseSensitive:(BOOL)flag
```

Parameters

sender

Panel requesting the ordering.

fileName1

String representing the first filename to order.

fileName2

String representing the second filename to order.

flag

If `YES`, the ordering is case-sensitive; if `NO`, it is not.

Return Value
One of the following:

- `NSOrderedAscending` if *fileName1* should precede *fileName2*
- `NSOrderedSame` if the two names are equivalent
- `NSOrderedDescending` if *fileName2* should precede *fileName1*

Discussion
Don't reorder filenames in the Save panel without good reason, because it may confuse the user to have files in one Save panel or Open panel ordered differently than those in other such panels or in the Finder. The default behavior of Save and Open panels is to order files as they appear in the Finder. Note also that by implementing this method you will reduce the operating performance of the panel.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

Declared In
`NSSavePanel.h`

panel:directoryDidChange:

Tells the delegate that the user has changed the selected directory in the `NSSavePanel` object specified. (Deprecated in Mac OS X v10.6. Use `panel:didChangeToDirectoryURL:` (page 3736) (`NSOpenSavePanelDelegate`) instead.)

```
- (void)panel:(id)sender directoryDidChange:(NSString *)path
```

Parameters

sender

Panel whose directory has changed.

path

String representing the new directory's path.

Availability
Available in Mac OS X v10.3 and later.
Deprecated in Mac OS X v10.6.

See Also
- `panel:didChangeToDirectoryURL:` (page 3736) (`NSOpenSavePanelDelegate`)

Declared In
`NSSavePanel.h`

panel:isValidFilename:

Gives the delegate the opportunity to validate selected items. (Deprecated in Mac OS X v10.6. Use `panel:validateURL:error:` (page 3737) (`NSOpenSavePanelDelegate`) instead. If both methods are implemented, the URL version will be called.)

```
- (BOOL)panel:(id)sender isValidFilename:(NSString *)filename
```

Parameters

sender
Panel requesting filename validation.

filename
String representing the filename to validate.

Return Value
`YES` if the filename is valid, or `NO` if the save panel should stay in its modal loop and wait for the user to type in or select a different filename or names.

Discussion
The `NSSavePanel` object *sender* sends this message just before the end of a modal session for each filename displayed or selected (including filenames in multiple selections). If the delegate refuses a filename in a multiple selection, none of the filenames in the selection is accepted.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `panel:validateURL:error:` (page 3737) (`NSOpenSavePanelDelegate`)

Declared In
`NSSavePanel.h`

panel:shouldShowFilename:

Gives the delegate the opportunity to filter items that it doesn't want the user to choose. (Deprecated in Mac OS X v10.6. Use `panel:shouldEnableURL:` (page 3736) (`NSOpenSavePanelDelegate`).)

```
- (BOOL)panel:(id)sender shouldShowFilename:(NSString *)filename
```

Parameters

sender
Panel that is querying whether it should show a certain file.

filename
String representing the name of the file to be loaded in the browser.

Return Value
`YES` if *`filename`* should be selectable, and `NO` if the save panel should disable the file or directory.

Discussion
The `NSSavePanel` object *`sender`* sends this message to the panel's delegate for each file or directory (filename) it is about to load in the browser.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
`- panel:shouldEnableURL:` (page 3736) (`NSOpenSavePanelDelegate`).

Declared In
`NSSavePanel.h`

Constants

Button tags

Button tags that refer to items on the panel.

```
enum {
   NSFileHandlingPanelCancelButton = NSCancelButton,
   NSFileHandlingPanelOKButton = NSOKButton
};
```

Constants
`NSFileHandlingPanelCancelButton`
The Cancel button.
Available in Mac OS X v10.0 and later.
Declared in `NSSavePanel.h`.

`NSFileHandlingPanelOKButton`
The OK button.
Available in Mac OS X v10.0 and later.
Declared in `NSSavePanel.h`.

`NSFileHandlingPanelImageButton`
Deleted in Mac OS X v10.3.
Available in Mac OS X v10.0 through Mac OS X v10.2.
Declared in `NSSavePanel.h`.

`NSFileHandlingPanelTitleField`
Deleted in Mac OS X v10.3.
Available in Mac OS X v10.0 through Mac OS X v10.2.
Declared in `NSSavePanel.h`.

`NSFileHandlingPanelBrowser`
Deleted in Mac OS X v10.3.
Available in Mac OS X v10.0 through Mac OS X v10.2.
Declared in `NSSavePanel.h`.

`NSFileHandlingPanelForm`
Deleted in Mac OS X v10.3.
Available in Mac OS X v10.0 through Mac OS X v10.2.
Declared in `NSSavePanel.h`.

`NSFileHandlingPanelHomeButton`
Deleted in Mac OS X v10.3.
Available in Mac OS X v10.0 through Mac OS X v10.2.
Declared in `NSSavePanel.h`.

`NSFileHandlingPanelDiskButton`
Deleted in Mac OS X v10.3.
Available in Mac OS X v10.0 through Mac OS X v10.2.
Declared in `NSSavePanel.h`.

`NSFileHandlingPanelDiskEjectButton`
Deleted in Mac OS X v10.3.
Available in Mac OS X v10.0 through Mac OS X v10.2.
Declared in `NSSavePanel.h`.

Declared In
`NSSavePanel.h`

NSScreen Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSScreen.h
Companion guide	Cocoa Drawing Guide
Related sample code	DesktopImage FunHouse GLUT OpenCL NBody Simulation Example UIElementInspector

Overview

An `NSScreen` object describes the attributes of a computer's monitor, or screen. An application may use an `NSScreen` object to retrieve information about a screen and use this information to decide what to display upon that screen. For example, an application may use the `deepestScreen` (page 2314) method to find out which of the available screens can best represent color and then may choose to display all of its windows on that screen.

The application object should be created before you use the methods in this class, so that the application object can make the necessary connection to the window system. You can make sure the application object exists by invoking the `sharedApplication` (page 135) method of `NSApplication`. If you created your application with Xcode, the application object is automatically created for you during initialization.

Note: The `NSScreen` class is for getting information about the available displays only. If you need additional information or want to change the attributes relating to a display, you must use Quartz Services. For more information, see *Quartz Display Services Reference*.

Tasks

Getting NSScreen Objects

`+ mainScreen` (page 2315)
Returns the `NSScreen` object containing the window with the keyboard focus.

`+ deepestScreen` (page 2314)
Returns an `NSScreen` object representing the screen that can best represent color.

`+ screens` (page 2315)
Returns an array of `NSScreen` objects representing all of the screens available on the system.

Getting Screen Information

`- depth` (page 2316)
Returns the receiver's current bit depth and colorspace information.

`- frame` (page 2317)
Returns the dimensions and location of the receiver.

`- supportedWindowDepths` (page 2318)
Returns a zero-terminated array of the window depths supported by the receiver.

`- deviceDescription` (page 2317)
Returns the device dictionary for the screen.

`- userSpaceScaleFactor` (page 2318)
Returns the scaling factor from user space to device space on the screen represented by the receiver.

`- visibleFrame` (page 2318)
Returns the current location and dimensions of the visible screen.

`- colorSpace` (page 2316)
Returns the colorSpace of the screen

Class Methods

deepestScreen

Returns an `NSScreen` object representing the screen that can best represent color.

```
+ (NSScreen *)deepestScreen
```

Return Value
The screen with the highest bit depth.

Discussion
This method always returns an object, even if there is only one screen and it is not a color screen.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSScreen.h`

mainScreen

Returns the `NSScreen` object containing the window with the keyboard focus.

```
+ (NSScreen *)mainScreen
```

Return Value
The main screen object.

Discussion
The main screen is not necessarily the same screen that contains the menu bar or has its origin at (0, 0). The main screen refers to the screen containing the window that is currently receiving keyboard events. It is the main screen because it is the one with which the user is most likely interacting.

The screen containing the menu bar is always the first object (index 0) in the array returned by the `screens` method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ screens` (page 2315)

Related Sample Code
ColorMatching
FunHouse
LightTable
MyPhoto
OpenCL NBody Simulation Example

Declared In
`NSScreen.h`

screens

Returns an array of `NSScreen` objects representing all of the screens available on the system.

```
+ (NSArray *)screens
```

Return Value
An array of the `NSScreen` objects available on the current system or `nil` if the screen information could not be obtained from the window system.

Discussion
The screen at index 0 in the returned array corresponds to the primary screen of the user's system. This is the screen that contains the menu bar and whose origin is at the point (0, 0). In the case of mirroring, the first screen is the largest drawable display; if all screens are the same size, it is the screen with the highest pixel depth. This primary screen may not be the same as the one returned by the `mainScreen` (page 2315) method, which returns the screen with the active window.

The array should not be cached. Screens can be added, removed, or dynamically reconfigured at any time. When the display configuration is changed, the default notification center sends a `NSApplicationDidChangeScreenParametersNotification` (page 194) notification.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AnimatedTableView
DesktopImage
GLUT

Declared In
`NSScreen.h`

Instance Methods

colorSpace

Returns the colorSpace of the screen

```
- (NSColorSpace *)colorSpace
```

Return Value
The colorSpace of the screen.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSScreen.h`

depth

Returns the receiver's current bit depth and colorspace information.

```
- (NSWindowDepth)depth
```

Return Value
The window depth information. This value cannot be used directly. You must pass it to a function such as `NSBitsPerPixelFromDepth` (page 3966) or `NSColorSpaceFromDepth` (page 3966) to obtain a concrete value for the desired information.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSScreen.h`

deviceDescription

Returns the device dictionary for the screen.

```
- (NSDictionary *)deviceDescription
```

Return Value
A dictionary containing the attributes of the receiver's screen. For the list of keys you can use to retrieve values from the returned dictionary, see `Display Deviceâ Descriptions` (page 3413).

Discussion
In addition to the display device constants described in *NSWindow Class Reference*, you can also retrieve the `CGDirectDisplayID` value associated with the screen from this dictionary. To access this value, specify the Objective-C string `@"NSScreenNumber"` as the key when requesting the item from the dictionary. The value associated with this key is an `NSNumber` object containing the display ID value. This string is only valid when used as a key for the dictionary returned by this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSScreen.h`

frame

Returns the dimensions and location of the receiver.

```
- (NSRect)frame
```

Return Value
The full screen rectangle at the current resolution. This rectangle includes any space currently occupied by the menu bar and dock.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- visibleFrame` (page 2318)

Related Sample Code
ColorMatching
FunHouse

GLUT
LightTable
UIElementInspector

Declared In
NSScreen.h

supportedWindowDepths

Returns a zero-terminated array of the window depths supported by the receiver.

```
- (const NSWindowDepth *)supportedWindowDepths
```

Return Value
A C-style array of window depths. The returned values cannot be used directly. You must pass each one to a function such as `NSBitsPerPixelFromDepth` (page 3966) or `NSColorSpaceFromDepth` (page 3966) to obtain a concrete value for the desired screen.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSScreen.h

userSpaceScaleFactor

Returns the scaling factor from user space to device space on the screen represented by the receiver.

```
- (CGFloat)userSpaceScaleFactor
```

Return Value
The scaling factor, measured in pixels per point, where a point is always equal to 1/72 of an inch. For example, a scaling factor of 2.0 indicates the display has a resolution 2 pixels per point or 144 pixels-per-inch.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSScreen.h

visibleFrame

Returns the current location and dimensions of the visible screen.

```
- (NSRect)visibleFrame
```

Return Value
The rectangle defining the portion of the screen in which it is currently safe to draw your application content.

Discussion
The returned rectangle is always based on the current user-interface settings and does not include the area currently occupied by the dock and menu bar. Because it is based on the current user -interface settings, the returned rectangle can change between calls and should not be cached.

Note: Even when dock hiding is enabled, the rectangle returned by this method may be smaller than the full screen. The system uses a small boundary area to determine when it should display the dock.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- frame` (page 2317)

Related Sample Code
CocoaDVDPlayer
MyPhoto
OpenCL NBody Simulation Example
QTQuartzPlayer
RoundTransparentWindow

Declared In
`NSScreen.h`

Notifications

NSScreenColorSpaceDidChangeNotification

Posted when the colorSpace of the screen has changed.

The notification object is the `NSScreen` whose `colorSpace` has changed.. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSScreen.h`

NSScroller Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSScroller.h
Companion guide	Cocoa Drawing Guide
Related sample code	PDFKitLinker2 Quartz Composer WWDC 2005 TextEdit Sketch+Accessibility

Class at a Glance

An NSScroller object is a user control for scrolling a document view within a container view. You normally don't need to program with NSScrollers, as Interface Builder allows you to fully configure them with an NSScrollView.

Principal Attributes

- Scrolling by small and large increments
- Proportional knob showing visible amount of document

Commonly Used Methods

`hitPart` (page 2328)
Indicates where the user clicked the NSScroller.

`floatValue` (page 819) (NSControl)
Returns the position of the NSScroller's knob.

`setFloatValue:knobProportion:` (page 2330)
Sets the position and size of the NSScroller's knob.

Overview

An `NSScroller` object controls scrolling of a document view within an NSScrollView's clip view (or potentially another kind of container view). It typically displays a pair of buttons that the user can click to scroll by a small amount (called a line increment or decrement) and Alt-click to scroll by a large amount (called a page increment or decrement), plus a slot containing a knob that the user can drag directly to the desired location. The knob indicates both the position within the document view and, by varying in size within the slot, the amount visible relative to the size of the document view. You can configure whether an `NSScroller` object uses scroll buttons, but it always draws the knob when there's room for it.

Don't use an NSScroller when an NSSlider would be better. A slider represents a range of values for something in the application and lets the user choose a setting. A scroller represents the relative position of the visible portion of a view and lets the user choose which portion to view.

Tasks

Determining NSScroller Size

`+ scrollerWidth` (page 2324)
Returns the width of "normal-sized" instances.

`+ scrollerWidthForControlSize:` (page 2324)
Returns the width of the scroller based on *`controlSize`*.

`- setControlSize:` (page 2329)
Sets the size of the receiver.

`- controlSize` (page 2325)
Returns the size of the receiver.

Laying out an NSScroller

`- setArrowsPosition:` (page 2329)
Sets the location of the scroll buttons within the receiver to *`location`*, or inhibits their display.

`- arrowsPosition` (page 2325)
Returns the location of the scroll buttons within the receiver, as described in `NSScrollArrowPosition` (page 2334).

Setting the Knob Position

`- setKnobProportion:` (page 2331)
Sets the proportion of the knob slot the knob should fill.

`- knobProportion` (page 2328)
Returns the portion of the knob slot the knob should fill, as a floating-point value from 0.0 (minimal size) to 1.0 (fills the slot).

- `setFloatValue:knobProportion:` (page 2330) Deprecated in Mac OS X v10.5
  Sets the position of the knob to *aFloat*, which is a value from 0.0 (indicating the top or left end) to 1.0 (the bottom or right end). (Deprecated. Code that targets Mac OS X 10.5 and later should use `setKnobProportion:` (page 2331) and `setDoubleValue:` (page 831).)

Calculating Layout

- `rectForPart:` (page 2328)
  Returns the rectangle occupied by *aPart*, which for this method is interpreted literally rather than as an indicator of scrolling direction.
- `testPart:` (page 2331)
  Returns the part that would be hit by a mouse-down event at *aPoint* (expressed in the window's coordinate system).
- `checkSpaceForParts` (page 2325)
  Checks to see if there is enough room in the receiver to display the knob and buttons.
- `usableParts` (page 2332)
  Returns a value indicating which parts of the receiver are displayed and usable.

Drawing the Parts

- `drawArrow:highlight:` (page 2326)
  Draws the scroll button indicated by *arrow*, which is either `NSScrollerIncrementArrow` (the down or right scroll button) or `NSScrollerDecrementArrow` (up or left).
- `drawKnobSlotInRect:highlight:` (page 2327)
  Draws the portion of the scroller's track, possibly including the line increment and decrement arrow buttons, that falls in the given slotRect.
- `drawKnob` (page 2326)
  Draws the knob.
- `drawParts` (page 2327)
  Caches images for the scroll buttons and knob.
- `highlight:` (page 2327)
  Highlights or unhighlights the scroll button the user clicked.

Event Handling

- `hitPart` (page 2328)
  Returns a part code indicating the manner in which the scrolling should be performed.
- `trackKnob:` (page 2331)
  Tracks the knob and sends action messages to the receiver's target.
- `trackScrollButtons:` (page 2332)
  Tracks the scroll buttons and sends action messages to the receiver's target.

Setting Control Tint

- `controlTint` (page 2326)
  Returns the receiver's control tint.
- `setControlTint:` (page 2330)
  Sets the receiver's control tint.

Class Methods

scrollerWidth

Returns the width of "normal-sized" instances.

```
+ (CGFloat)scrollerWidth
```

Discussion
NSScrollView uses this value to lay out its components. Subclasses that use a different width should override this method.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
PDFKitLinker2

Declared In
`NSScroller.h`

scrollerWidthForControlSize:

Returns the width of the scroller based on *controlSize*.

```
+ (CGFloat)scrollerWidthForControlSize:(NSControlSize)controlSize
```

Discussion
Valid values for *controlSize* are described in NSCell's `NSControlSize` (page 620).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSScroller.h`

Instance Methods

arrowsPosition

Returns the location of the scroll buttons within the receiver, as described in `NSScrollArrowPosition` (page 2334).

```
- (NSScrollArrowPosition)arrowsPosition
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setArrowsPosition:` (page 2329)

Declared In
`NSScroller.h`

checkSpaceForParts

Checks to see if there is enough room in the receiver to display the knob and buttons.

```
- (void)checkSpaceForParts
```

Discussion
`usableParts` (page 2332) returns the state calculated by this method. You should never need to invoke this method; it's invoked automatically whenever the NSScroller's size changes.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSScroller.h`

controlSize

Returns the size of the receiver.

```
- (NSControlSize)controlSize
```

Discussion
Valid return values are described in `NSControlSize` (page 620).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setControlSize:` (page 2329)

Declared In
`NSScroller.h`

controlTint

Returns the receiver's control tint.

```
- (NSControlTint)controlTint
```

Discussion
Valid return values are described in `NSControlTint` (page 619).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setControlTint:` (page 2330)

Declared In
`NSScroller.h`

drawArrow:highlight:

Draws the scroll button indicated by *`arrow`*, which is either `NSScrollerIncrementArrow` (the down or right scroll button) or `NSScrollerDecrementArrow` (up or left).

```
- (void)drawArrow:(NSScrollerArrow)arrow highlight:(BOOL)flag
```

Discussion
If *`flag`* is `YES`, the button is drawn highlighted; otherwise it's drawn normally. You should never need to invoke this method directly, but may wish to override it to customize the appearance of scroll buttons.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawKnob` (page 2326)
- `rectForPart:` (page 2328)

Declared In
`NSScroller.h`

drawKnob

Draws the knob.

```
- (void)drawKnob
```

Discussion
You should never need to invoke this method directly, but may wish to override it to customize the appearance of the knob.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawArrow:highlight:` (page 2326)

- rectForPart: (page 2328)

Declared In
NSScroller.h

drawKnobSlotInRect:highlight:

Draws the portion of the scroller's track, possibly including the line increment and decrement arrow buttons, that falls in the given slotRect.

```
- (void)drawKnobSlotInRect:(NSRect)slotRect highlight:(BOOL)flag
```

Parameters

slotRect

The rectangle in which to draw the knob slot.

flag

If `flag` is `YES`, any scroll arrow button that falls within *slotRect* is drawn highlighted; otherwise it's drawn normally.

Discussion
Only one arrow button will be shown highlighted at a time, so you can expect this method to sometimes be invoked with a slotRect that encompasses only one arrow.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSScroller.h

drawParts

Caches images for the scroll buttons and knob.

```
- (void)drawParts
```

Discussion
It's invoked only once when the NSScroller is created. You may want to override this method if you alter the look of the NSScroller, but you should never invoke it directly.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSScroller.h

highlight:

Highlights or unhighlights the scroll button the user clicked.

```
- (void)highlight:(BOOL)flag
```

Discussion
The receiver invokes this method while tracking the mouse; you should not invoke it directly. If *flag* is YES, the appropriate part is drawn highlighted; otherwise it's drawn normally.

Availability
Available in Mac OS X v10.0 and later.

See Also
- drawArrow:highlight: (page 2326)
- rectForPart: (page 2328)

Declared In
NSScroller.h

hitPart

Returns a part code indicating the manner in which the scrolling should be performed.

```
- (NSScrollerPart)hitPart
```

Discussion
See NSScrollerPart (page 2332) for a list of part codes.

This method is typically invoked by an NSScrollView to determine how to scroll its document view when it receives an action message from the NSScroller.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSScroller.h

knobProportion

Returns the portion of the knob slot the knob should fill, as a floating-point value from 0.0 (minimal size) to 1.0 (fills the slot).

```
- (CGFloat)knobProportion
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSScroller.h

rectForPart:

Returns the rectangle occupied by *aPart*, which for this method is interpreted literally rather than as an indicator of scrolling direction.

```
- (NSRect)rectForPart:(NSScrollerPart)aPart
```

Discussion
See `NSScrollerPart` (page 2332) for a list of possible values for *`aPart`*.

Note the interpretations of `NSScrollerDecrementPage` and `NSScrollerIncrementPage`. The actual part of an NSScroller that causes page-by-page scrolling varies, so as a convenience these part codes refer to useful parts different from the scroll buttons.

Returns `NSZeroRect` if the part requested isn't present on the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hitPart` (page 2328)
- `testPart:` (page 2331)
- `usableParts` (page 2332)

Declared In
`NSScroller.h`

setArrowsPosition:

Sets the location of the scroll buttons within the receiver to *`location`*, or inhibits their display.

```
- (void)setArrowsPosition:(NSScrollArrowPosition)location
```

Discussion
See `NSScrollArrowPosition` (page 2334) for a list of possible values for *`location`*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `arrowsPosition` (page 2325)

Declared In
`NSScroller.h`

setControlSize:

Sets the size of the receiver.

```
- (void)setControlSize:(NSControlSize)controlSize
```

Discussion
Valid values for *`controlSize`* are described in `NSControlSize` (page 620).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `controlSize` (page 2325)

Related Sample Code
FunHouse
Quartz Composer QCTV
WhackedTV

Declared In
NSScroller.h

setControlTint:

Sets the receiver's control tint.

```
- (void)setControlTint:(NSControlTint)controlTint
```

Discussion
Valid values for *controlTint* are described in NSControlTint (page 619).

Availability
Available in Mac OS X v10.0 and later.

See Also
- controlTint (page 2326)

Declared In
NSScroller.h

setFloatValue:knobProportion:

Sets the position of the knob to *aFloat*, which is a value from 0.0 (indicating the top or left end) to 1.0 (the bottom or right end). (Deprecated in Mac OS X v10.5. Code that targets Mac OS X 10.5 and later should use setKnobProportion: (page 2331) and setDoubleValue: (page 831).)

```
- (void)setFloatValue:(float)aFloat knobProportion:(CGFloat)knobProp
```

Discussion
Also sets the proportion of the knob slot filled by the knob to *knobProp*, also a value from 0.0 (minimal size) to 1.0 (fills the slot).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

See Also
- floatValue (page 819) (NSControl)
- knobProportion (page 2328)

Declared In
NSScroller.h

setKnobProportion:

Sets the proportion of the knob slot the knob should fill.

```
- (void)setKnobProportion:(CGFloat)proportion
```

Parameters

proportion

A floating point value between 0.0 (minimal size) and 1.0 (fills the entire slot).

Discussion

Code that targets Mac OS X 10.5 and later should use -`setKnobProportion:` (page 2331) and `setDoubleValue:` (page 831): in preference to the deprecated method that they replace, `setFloatValue:knobProportion:` (page 2330). These methods provide more uniform, Key Value Coding-compatible access to the two values, and allow for a double-precision scroll position.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSScroller.h`

testPart:

Returns the part that would be hit by a mouse-down event at *aPoint* (expressed in the window's coordinate system).

```
- (NSScrollerPart)testPart:(NSPoint)aPoint
```

Discussion

See `NSScrollerPart` (page 2332) for a list of possible return values.

Note the interpretations of `NSScrollerDecrementPage` and `NSScrollerIncrementPage`. The actual part of an NSScroller that causes page-by-page scrolling varies, so as a convenience these part codes refer to useful parts different from the scroll buttons.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `hitPart` (page 2328)
- `rectForPart:` (page 2328)

Declared In

`NSScroller.h`

trackKnob:

Tracks the knob and sends action messages to the receiver's target.

```
- (void)trackKnob:(NSEvent *)theEvent
```

Discussion
This method is invoked automatically when the receiver receives *theEvent* mouse-down event in the knob; you should not invoke it directly.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSScroller.h

trackScrollButtons:

Tracks the scroll buttons and sends action messages to the receiver's target.

```
- (void)trackScrollButtons:(NSEvent *)theEvent
```

Discussion
This method is invoked automatically when the receiver receives *theEvent* mouse-down event in a scroll button; you should not invoke this method directly.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSScroller.h

usableParts

Returns a value indicating which parts of the receiver are displayed and usable.

```
- (NSUsableScrollerParts)usableParts
```

Discussion
See NSUsableScrollerParts (page 2335) for a list of possible values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- checkSpaceForParts (page 2325)
- arrowsPosition (page 2325)

Declared In
NSScroller.h

Constants

NSScrollerPart

These constants specify the different parts of the scroller:

```
typedef enum _NSScrollerPart {
    NSScrollerNoPart = 0,
    NSScrollerDecrementPage = 1,
    NSScrollerKnob = 2,
    NSScrollerIncrementPage = 3,
    NSScrollerDecrementLine = 4,
    NSScrollerIncrementLine = 5,
    NSScrollerKnobSlot = 6
} NSScrollerPart;
```

Constants

`NSScrollerKnob`
Directly to the NSScroller's value, as given by `floatValue` (page 819).
Available in Mac OS X v10.0 and later.
Declared in `NSScroller.h`.

`NSScrollerKnobSlot`
Directly to the NSScroller's value, as given by `floatValue` (page 819).
Available in Mac OS X v10.0 and later.
Declared in `NSScroller.h`.

`NSScrollerDecrementLine`
Up or left by a small amount.
Available in Mac OS X v10.0 and later.
Declared in `NSScroller.h`.

`NSScrollerDecrementPage`
Up or left by a large amount.
Available in Mac OS X v10.0 and later.
Declared in `NSScroller.h`.

`NSScrollerIncrementLine`
Down or right by a small amount.
Available in Mac OS X v10.0 and later.
Declared in `NSScroller.h`.

`NSScrollerIncrementPage`
Down or right by a large amount.
Available in Mac OS X v10.0 and later.
Declared in `NSScroller.h`.

`NSScrollerNoPart`
Don't scroll at all.
Available in Mac OS X v10.0 and later.
Declared in `NSScroller.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSScroller.h`

NSScrollerArrow

These constants describe the two scroller buttons and are used by `drawArrow:highlight:` (page 2326):

```
typedef enum _NSScrollerArrow {
    NSScrollerIncrementArrow = 0,
    NSScrollerDecrementArrow = 1
} NSScrollerArrow;
```

Constants

`NSScrollerIncrementArrow`
The down or right scroll button.

Available in Mac OS X v10.0 and later.

Declared in `NSScroller.h`.

`NSScrollerDecrementArrow`
The up or left scroll button.

Available in Mac OS X v10.0 and later.

Declared in `NSScroller.h`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSScroller.h`

NSScrollArrowPosition

These constants specify where the scroller's buttons appear and are used by `arrowsPosition` (page 2325) and `setArrowsPosition:` (page 2329).

```
typedef enum _NSScrollArrowPosition {
    NSScrollerArrowsMaxEnd            = 0,
    NSScrollerArrowsMinEnd            = 1,
    NSScrollerArrowsDefaultSetting    = 0,
    NSScrollerArrowsNone              = 2
} NSScrollArrowPosition;
```

Constants

`NSScrollerArrowsMaxEnd`
Buttons at bottom or right. This constant has been deprecated.

Available in Mac OS X v10.0 and later.

Declared in `NSScroller.h`.

`NSScrollerArrowsMinEnd`
Buttons at top or left. This has been deprecated.

Available in Mac OS X v10.0 and later.

Declared in `NSScroller.h`.

`NSScrollerArrowsDefaultSetting`
Buttons are displayed according to the system-wide appearance preferences.

Available in Mac OS X v10.1 and later.

Declared in `NSScroller.h`.

`NSScrollerArrowsNone`
No buttons.

Available in Mac OS X v10.0 and later.

Declared in `NSScroller.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSScroller.h`

NSUsableScrollerParts

These constants specify which parts of the scroller are visible.

```
typedef enum _NSUsableScrollerParts {
    NSNoScrollerParts = 0,
    NSOnlyScrollerArrows = 1,
    NSAllScrollerParts = 2
} NSUsableScrollerParts;
```

Constants
`NSNoScrollerParts`
Scroller has neither a knob nor scroll buttons, only the knob slot.

Available in Mac OS X v10.0 and later.

Declared in `NSScroller.h`.

`NSOnlyScrollerArrows`
Scroller has only scroll buttons, no knob.

Available in Mac OS X v10.0 and later.

Declared in `NSScroller.h`.

`NSAllScrollerParts`
Scroller has at least a knob, possibly also scroll buttons.

Available in Mac OS X v10.0 and later.

Declared in `NSScroller.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSScroller.h`

NSScrollView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSScrollView.h
Companion guide	Scroll View Programming Guide for Cocoa
Related sample code	Quartz Composer WWDC 2005 TextEdit Rulers Sketch-112 TextSizingExample WhackedTV

Class at a Glance

An `NSScrollView` allows the user to scroll a document view that's too large to display in its entirety. In addition to the document view, it displays horizontal and vertical scrollers and rulers (depending on which it's configured to have).

Principal Attributes

- Configurable scrollers
- Configurable rulers
- Small and large increment scrolling
- Dynamic (continuous) scrolling
- Display of a special cursor over its document view

Interface Builder
: Drag a scroll view to a window.

Commonly Used Methods

`setDocumentView:` (page 2355)
Sets the scroll view's document view.

`setLineScroll:` (page 2360)
Sets the amount by which the document view moves during scrolling.

`setRulersVisible:` (page 2361)
Displays or hides rulers.

Overview

The NSScrollView class is the central coordinator for the Application Kit's scrolling machinery, composed of this class, NSClipView, and NSScroller. An NSScrollView displays a portion of a document view that's too large to be displayed whole and provides NSScroller scroll bars that allow the user to move the document view within the NSScrollView. Note that, when using an NSClipView within an NSScrollView (the usual configuration), you should issue messages that control background drawing state to the NSScrollView, rather than messaging the NSClipView directly.

Tasks

Calculating Layout

`+ contentSizeForFrameSize:hasHorizontalScroller:hasVerticalScroller:borderType:` (page 2342)
Returns the size of a content view for an NSScrollView whose frame size is *frameSize*.

`+ frameSizeForContentSize:hasHorizontalScroller:hasVerticalScroller:borderType:` (page 2342)
Returns the frame size of an NSScrollView that contains a content view whose size is *contentSize*.

Determining Component Sizes

`- contentSize` (page 2345)
Returns the size of the receiver's content view.

`- documentVisibleRect` (page 2346)
Returns the portion of the document view, in its own coordinate system, visible through the receiver's content view.

Managing Graphics Attributes

`- setBackgroundColor:` (page 2353)
Sets the color of the content view's background to *aColor*.

- `backgroundColor` (page 2344)
  Returns the content view's background color.
- `drawsBackground` (page 2347)
  Returns `YES` if the receiver cell fills the background with its background color; otherwise, `NO`.
- `setDrawsBackground:` (page 2356)
  Sets whether the receiver draws its background.
- `setBorderType:` (page 2354)
  Sets the border type of the receiver to *borderType*.
- `borderType` (page 2344)
  Returns a value that represents the type of border surrounding the receiver; see the description of `setBorderType:` (page 2354) for a list of possible values.
- `setDocumentCursor:` (page 2355)
  Sets the cursor used when the cursor is over the content view to *aCursor*, by sending `setDocumentCursor:` (page 2355) to the content view.
- `documentCursor` (page 2346)
  Returns the content view's document cursor.

Managing the Scrolled Views

- `setContentView:` (page 2354)
  Sets the receiver's content view, the view that clips the document view, to *aView*.
- `contentView` (page 2345)
  Returns the receiver's content view, the view that clips the document view.
- `setDocumentView:` (page 2355)
  Sets the receiver's document view to *aView*.
- `documentView` (page 2346)
  Returns the view the receiver scrolls within its content view.

Managing Scrollers

- `setHorizontalScroller:` (page 2360)
  Sets the receiver's horizontal scroller to *aScroller*, establishing the appropriate target-action relationships between them.
- `horizontalScroller` (page 2350)
  Returns the receiver's horizontal scroller, regardless of whether the receiver is currently displaying it, or `nil` if the receiver has none.
- `setHasHorizontalScroller:` (page 2357)
  Determines whether the receiver keeps a horizontal scroller.
- `hasHorizontalScroller` (page 2347)
  Returns `YES` if the receiver displays a horizontal scroller, `NO` if it doesn't.
- `setVerticalScroller:` (page 2364)
  Sets the receiver's vertical scroller to *aScroller*, establishing the appropriate target-action relationships between them.

- `verticalScroller` (page 2366)
  Returns the receiver's vertical scroller, regardless of whether the receiver is currently displaying it, or `nil` if the receiver has none.
- `setHasVerticalScroller:` (page 2358)
  Determines whether the receiver keeps a vertical scroller.
- `hasVerticalScroller` (page 2348)
  Returns `YES` if the receiver displays a vertical scroller, `NO` if it doesn't.
- `setAutohidesScrollers:` (page 2353)
  Determines whether the receiver automatically hides its scroll bars when they are not needed.
- `autohidesScrollers` (page 2344)
  Returns `YES` when autohiding is set for scroll bars in the receiver.

Managing Rulers

+ `setRulerViewClass:` (page 2343)
  Sets the default class to be used for ruler objects in NSScrollViews to *aClass*.
+ `rulerViewClass` (page 2343)
  Returns the default class to be used for ruler objects in NSScrollViews.
- `setHasHorizontalRuler:` (page 2356)
  Determines whether the receiver keeps a horizontal ruler object.
- `hasHorizontalRuler` (page 2347)
  Returns `YES` if the receiver maintains a horizontal ruler view, `NO` if it doesn't.
- `setHorizontalRulerView:` (page 2359)
  Sets the receiver's horizontal ruler view to *aRulerView*.
- `horizontalRulerView` (page 2350)
  Returns the receiver's horizontal ruler view, regardless of whether the receiver is currently displaying it, or `nil` if the receiver has none.
- `setHasVerticalRuler:` (page 2357)
  Determines whether the receiver keeps a vertical ruler object.
- `hasVerticalRuler` (page 2348)
  Returns `YES` if the receiver maintains a vertical ruler view, `NO` if it doesn't.
- `setVerticalRulerView:` (page 2363)
  Sets the receiver's vertical ruler view to *aRulerView*.
- `verticalRulerView` (page 2365)
  Returns the receiver's vertical ruler view, regardless of whether the receiver is currently displaying it, or `nil` if the receiver has none.
- `setRulersVisible:` (page 2361)
  Determines whether the receiver displays its rulers.
- `rulersVisible` (page 2352)
  Returns `YES` if the receiver was set to show rulers using `setRulersVisible:` (page 2361) (whether or not it has rulers at all), `NO` if it was set to hide them.

Setting Scrolling Behavior

- `setLineScroll:` (page 2360)
  Sets the horizontal and vertical line scroll amounts to *aFloat*.
- `lineScroll` (page 2350)
  Returns the vertical line scroll amount: the amount by which the receiver scrolls itself vertically when scrolling line by line, expressed in the content view's coordinate system.
- `setHorizontalLineScroll:` (page 2358)
  Sets the amount by which the receiver scrolls itself horizontally when scrolling line by line to *aFloat*, expressed in the content view's coordinate system.
- `horizontalLineScroll` (page 2349)
  Returns the amount by which the receiver scrolls itself horizontally when scrolling line by line, expressed in the content view's coordinate system.
- `setVerticalLineScroll:` (page 2362)
  Sets the amount by which the receiver scrolls itself vertically when scrolling line by line to *aFloat*, expressed in the content view's coordinate system.
- `verticalLineScroll` (page 2364)
  Returns the amount by which the receiver scrolls itself vertically when scrolling line by line, expressed in the content view's coordinate system.
- `setPageScroll:` (page 2361)
  Sets the horizontal and vertical page scroll amounts to *aFloat*.
- `pageScroll` (page 2351)
  Returns the vertical page scroll amount: the amount of the document view kept visible when scrolling vertically page by page, expressed in the content view's coordinate system.
- `setHorizontalPageScroll:` (page 2359)
  Sets the amount of the document view kept visible when scrolling horizontally page by page to *aFloat*, expressed in the content view's coordinate system.
- `horizontalPageScroll` (page 2349)
  Returns the amount of the document view kept visible when scrolling horizontally page by page, expressed in the content view's coordinate system.
- `setVerticalPageScroll:` (page 2363)
  Sets the amount of the document view kept visible when scrolling vertically page by page to *aFloat*, expressed in the content view's coordinate system.
- `verticalPageScroll` (page 2365)
  Returns the amount of the document view kept visible when scrolling vertically page by page, expressed in the content view's coordinate system.
- `setScrollsDynamically:` (page 2362)
  Determines whether the receiver redraws its document view while scrolling continuously.
- `scrollsDynamically` (page 2352)
  Returns `YES` if the receiver redraws its document view while tracking the knob, `NO` if it redraws only when the scroller knob is released.
- `scrollWheel:` (page 2353)
  Scrolls the receiver up or down, in response to the user moving the mouse's scroll wheel specified by *theEvent*.

Updating Display After Scrolling

`- reflectScrolledClipView:` (page 2351)
Adjusts the receiver's scrollers to reflect the size and positioning of its content view.

Arranging Components

`- tile` (page 2364)
Lays out the components of the receiver: the content view, the scrollers, and the ruler views.

Class Methods

contentSizeForFrameSize:hasHorizontalScroller:hasVerticalScroller:borderType:

Returns the size of a content view for an NSScrollView whose frame size is *frameSize*.

```
+ (NSSize)contentSizeForFrameSize:(NSSize)frameSize hasHorizontalScroller:(BOOL)hFlag
    hasVerticalScroller:(BOOL)vFlag borderType:(NSBorderType)borderType
```

Discussion
hFlag and *vFlag* indicate whether a horizontal or vertical scroller, respectively, is present. If either flag is `YES` then the content size is reduced in the appropriate dimension by the width of an NSScroller. The *borderType* argument indicates the appearance of the NSScrollView's edge, which also affects the content size; see the description of `setBorderType:` (page 2354) for a list of possible values.

For an existing NSScrollView, you can simply use the `contentSize` (page 2345) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ frameSizeForContentSize:hasHorizontalScroller:hasVerticalScroller:borderType:` (page 2342)
`+ scrollerWidth` (page 2324) (NSScroller)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSScrollView.h`

frameSizeForContentSize:hasHorizontalScroller:hasVerticalScroller:borderType:

Returns the frame size of an NSScrollView that contains a content view whose size is *contentSize*.

```
+ (NSSize)frameSizeForContentSize:(NSSize)contentSize
    hasHorizontalScroller:(BOOL)hFlag hasVerticalScroller:(BOOL)vFlag
    borderType:(NSBorderType)borderType
```

Discussion
The *hFlag* and *vFlag* arguments indicate whether a horizontal or vertical scroller, respectively, is present. If either flag is YES then the frame size is increased in the appropriate dimension by the width of an NSScroller. *borderType* indicates the appearance of the NSScrollView's edge, which also affects the frame size; see the description of `setBorderType:` (page 2354) for a list of possible values.

For an existing NSScrollView, you can simply use the `frame` method and extract its size.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ contentSizeForFrameSize:hasHorizontalScroller:hasVerticalScroller:borderType:` (page 2342)
`+ scrollerWidth` (page 2324) (NSScroller)

Related Sample Code
CocoaAUHost
QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit

Declared In
NSScrollView.h

rulerViewClass

Returns the default class to be used for ruler objects in NSScrollViews.

```
+ (Class)rulerViewClass
```

Discussion
This class is normally NSRulerView.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ setRulerViewClass:` (page 2343)

Declared In
NSScrollView.h

setRulerViewClass:

Sets the default class to be used for ruler objects in NSScrollViews to *aClass*.

```
+ (void)setRulerViewClass:(Class)aClass
```

Discussion
This class is normally NSRulerView, but you can use this method to set it to a custom subclass of NSRulerView.

This method simply sets a global variable private to NSScrollView. Subclasses of NSScrollView should override both this method and `rulerViewClass` (page 2343) to store their ruler view classes in private variables.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ rulerViewClass` (page 2343)

Declared In
`NSScrollView.h`

Instance Methods

autohidesScrollers

Returns `YES` when autohiding is set for scroll bars in the receiver.

```
- (BOOL)autohidesScrollers
```

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setAutohidesScrollers:` (page 2353)

Declared In
`NSScrollView.h`

backgroundColor

Returns the content view's background color.

```
- (NSColor *)backgroundColor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setBackgroundColor:` (page 2353)
`- backgroundColor` (page 632) (NSClipView)

Declared In
`NSScrollView.h`

borderType

Returns a value that represents the type of border surrounding the receiver; see the description of `setBorderType:` (page 2354) for a list of possible values.

```
- (NSBorderType)borderType
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit

Declared In
NSScrollView.h

contentSize

Returns the size of the receiver's content view.

```
- (NSSize)contentSize
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ contentSizeForFrameSize:hasHorizontalScroller:hasVerticalScroller:borderType:` (page 2342)

Related Sample Code
CustomSave
FunHouse
Quartz Composer WWDC 2005 TextEdit

Declared In
NSScrollView.h

contentView

Returns the receiver's content view, the view that clips the document view.

```
- (NSClipView *)contentView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setContentView:` (page 2354)
`- documentView` (page 2346)

Related Sample Code
FunHouse
QTAudioContextInsert
WhackedTV

Declared In
NSScrollView.h

documentCursor

Returns the content view's document cursor.

```
- (NSCursor *)documentCursor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDocumentCursor:` (page 2355)
- `documentCursor` (page 633) (NSClipView)

Declared In
`NSScrollView.h`

documentView

Returns the view the receiver scrolls within its content view.

```
- (id)documentView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDocumentView:` (page 2355)
- `documentView` (page 634) (NSClipView)

Related Sample Code
CocoaAUHost
QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit
Rulers

Declared In
`NSScrollView.h`

documentVisibleRect

Returns the portion of the document view, in its own coordinate system, visible through the receiver's content view.

```
- (NSRect)documentVisibleRect
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `documentVisibleRect` (page 634) (NSClipView)
- `visibleRect` (page 3245) (NSView)

Related Sample Code
STUCAuthoringDeviceCocoaSample

Declared In
NSScrollView.h

drawsBackground

Returns YES if the receiver cell fills the background with its background color; otherwise, NO.

```
- (BOOL)drawsBackground
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSScrollView.h

hasHorizontalRuler

Returns YES if the receiver maintains a horizontal ruler view, NO if it doesn't.

```
- (BOOL)hasHorizontalRuler
```

Discussion
Display of rulers is controlled using the setRulersVisible: (page 2361) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- horizontalRulerView (page 2350)
- setHasHorizontalRuler: (page 2356)
- hasVerticalRuler (page 2348)
+ rulerViewClass (page 2343)

Declared In
NSScrollView.h

hasHorizontalScroller

Returns YES if the receiver displays a horizontal scroller, NO if it doesn't.

```
- (BOOL)hasHorizontalScroller
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- horizontalScroller (page 2350)
- setHasHorizontalScroller: (page 2357)

- hasVerticalScroller (page 2348)

Related Sample Code
QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility

Declared In
NSScrollView.h

hasVerticalRuler

Returns YES if the receiver maintains a vertical ruler view, NO if it doesn't.

```
- (BOOL)hasVerticalRuler
```

Discussion
Display of rulers is controlled using the setRulersVisible: (page 2361) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- verticalRulerView (page 2365)
- setHasVerticalRuler: (page 2357)
- hasHorizontalRuler (page 2347)
+ rulerViewClass (page 2343)

Declared In
NSScrollView.h

hasVerticalScroller

Returns YES if the receiver displays a vertical scroller, NO if it doesn't.

```
- (BOOL)hasVerticalScroller
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- verticalScroller (page 2366)
- setHasVerticalScroller: (page 2358)
- hasHorizontalScroller (page 2347)

Related Sample Code
QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit

Declared In
NSScrollView.h

horizontalLineScroll

Returns the amount by which the receiver scrolls itself horizontally when scrolling line by line, expressed in the content view's coordinate system.

```
- (CGFloat)horizontalLineScroll
```

Discussion
This amount is used when the user clicks the scroll arrows on the horizontal scroll bar without holding down a modifier key.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHorizontalLineScroll:` (page 2358)
- `verticalLineScroll` (page 2364)
- `setLineScroll:` (page 2360)
- `horizontalPageScroll` (page 2349)

Declared In
`NSScrollView.h`

horizontalPageScroll

Returns the amount of the document view kept visible when scrolling horizontally page by page, expressed in the content view's coordinate system.

```
- (CGFloat)horizontalPageScroll
```

Discussion
This amount is used when the user clicks the scroll arrows on the horizontal scroll bar while holding down the Option key.

This amount expresses the context that remains when the receiver scrolls by one page, allowing the user to orient to the new display. It differs from the line scroll amount, which indicates how far the document view moves. The page scroll amount is the amount common to the content view before and after the document view is scrolled by one page.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHorizontalPageScroll:` (page 2359)
- `verticalPageScroll` (page 2365)
- `setPageScroll:` (page 2361)
- `horizontalLineScroll` (page 2349)

Declared In
`NSScrollView.h`

horizontalRulerView

Returns the receiver's horizontal ruler view, regardless of whether the receiver is currently displaying it, or `nil` if the receiver has none.

```
- (NSRulerView *)horizontalRulerView
```

Discussion
If the receiver is set to display a horizontal ruler view and doesn't yet have one, this method creates an instance of the ruler view class set using the class method `setRulerViewClass:` (page 2343). Display of rulers is controlled using the `setRulersVisible:` (page 2361) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hasHorizontalRuler` (page 2347)
- `verticalRulerView` (page 2365)

Related Sample Code
Rulers
Sketch-112

Declared In
`NSScrollView.h`

horizontalScroller

Returns the receiver's horizontal scroller, regardless of whether the receiver is currently displaying it, or `nil` if the receiver has none.

```
- (NSScroller *)horizontalScroller
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Sketch+Accessibility

Declared In
`NSScrollView.h`

lineScroll

Returns the vertical line scroll amount: the amount by which the receiver scrolls itself vertically when scrolling line by line, expressed in the content view's coordinate system.

```
- (CGFloat)lineScroll
```

Discussion
This amount is used when the user clicks the scroll arrows on the vertical scroll bar without holding down a modifier key. As part of its implementation, this method calls `verticalLineScroll` (page 2364).

Note that a scroll view can have two different line scroll amounts: `verticalLineScroll` (page 2364) and `horizontalLineScroll` (page 2349). Use this method only if you can be sure they're both the same; for example, you always use `setLineScroll:` (page 2360), which sets both amounts to the same value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setLineScroll:` (page 2360)
- `verticalPageScroll` (page 2365)
- `horizontalPageScroll` (page 2349)

Declared In
`NSScrollView.h`

pageScroll

Returns the vertical page scroll amount: the amount of the document view kept visible when scrolling vertically page by page, expressed in the content view's coordinate system.

```
- (CGFloat)pageScroll
```

Discussion
This amount is used when the user clicks the scroll arrows on the vertical scroll bar while holding down the Option key. As part of its implementation, this method calls `verticalPageScroll` (page 2365).

This amount expresses the context that remains when the receiver scrolls by one page, allowing the user to orient to the new display. It differs from the line scroll amount, which indicates how far the document view moves. The page scroll amount is the amount common to the content view before and after the document view is scrolled by one page.

Note that a scroll view can have two different page scroll amounts: `verticalPageScroll` (page 2365) and `horizontalPageScroll` (page 2349). Use this method only if you can be sure they're both the same; for example, you always use `setPageScroll:` (page 2361), which sets both amounts to the same value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPageScroll:` (page 2361)
- `verticalLineScroll` (page 2364)
- `horizontalLineScroll` (page 2349)

Declared In
`NSScrollView.h`

reflectScrolledClipView:

Adjusts the receiver's scrollers to reflect the size and positioning of its content view.

```
- (void)reflectScrolledClipView:(NSClipView *)aClipView
```

Parameters

aClipView

The clip view being adjusted to. If *aClipView* is any view object other than the receiver's content view, the method does nothing.

Discussion

This method is invoked automatically during scrolling and when an `NSClipView` object's relationship to its document view changes; you should rarely need to invoke it yourself, but may wish to override it for custom updating or other behavior. If you override this method, be sure to call the superclass implementation. If you do not, other controls (such as the current scrollers) may not be updated properly.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `contentView` (page 2345)
- `documentView` (page 2346)

Related Sample Code

WhackedTV

Declared In

`NSScrollView.h`

rulersVisible

Returns `YES` if the receiver was set to show rulers using `setRulersVisible:` (page 2361) (whether or not it has rulers at all), `NO` if it was set to hide them.

```
- (BOOL)rulersVisible
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- `hasHorizontalRuler` (page 2347)
- `hasVerticalRuler` (page 2348)

Related Sample Code

Sketch+Accessibility

Sketch-112

Declared In

`NSScrollView.h`

scrollsDynamically

Returns `YES` if the receiver redraws its document view while tracking the knob, `NO` if it redraws only when the scroller knob is released.

```
- (BOOL)scrollsDynamically
```

Discussion
NSScrollView scrolls dynamically by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setScrollsDynamically:` (page 2362)

Declared In
`NSScrollView.h`

scrollWheel:

Scrolls the receiver up or down, in response to the user moving the mouse's scroll wheel specified by *theEvent*.

```
- (void)scrollWheel:(NSEvent *)theEvent
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSScrollView.h`

setAutohidesScrollers:

Determines whether the receiver automatically hides its scroll bars when they are not needed.

```
- (void)setAutohidesScrollers:(BOOL)flag
```

Discussion
The horizontal and vertical scroll bars are hidden independently of each other. When autohiding is on and the content of the receiver doesn't extend beyond the size of the clip view on a given axis, the scroller on that axis is removed to leave more room for the content.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `autohidesScrollers` (page 2344)

Related Sample Code
FunHouse
WhackedTV

Declared In
`NSScrollView.h`

setBackgroundColor:

Sets the color of the content view's background to *aColor*.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Discussion
This color is used to paint areas inside the content view that aren't covered by the document view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `backgroundColor` (page 2344)
- `setBackgroundColor:` (page 635) (NSClipView)

Declared In
`NSScrollView.h`

setBorderType:

Sets the border type of the receiver to *borderType*.

```
- (void)setBorderType:(NSBorderType)borderType
```

Discussion
borderType may be one of:

```
NSNoBorder
NSLineBorder
NSBezelBorder
NSGrooveBorder
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `borderType` (page 2344)

Related Sample Code
CustomSave
FunHouse
Sketch-112

Declared In
`NSScrollView.h`

setContentView:

Sets the receiver's content view, the view that clips the document view, to *aView*.

```
- (void)setContentView:(NSClipView *)aView
```

Discussion
If *aView* has a document view, this method also sets the receiver's document view to be the document view of *aView*. The original content view retains its document view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `contentView` (page 2345)
- `setDocumentView:` (page 2355)

Declared In
`NSScrollView.h`

setDocumentCursor:

Sets the cursor used when the cursor is over the content view to *aCursor*, by sending `setDocumentCursor:` (page 2355) to the content view.

```
- (void)setDocumentCursor:(NSCursor *)aCursor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `documentCursor` (page 2346)

Declared In
`NSScrollView.h`

setDocumentView:

Sets the receiver's document view to *aView*.

```
- (void)setDocumentView:(NSView *)aView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `documentView` (page 2346)
- `setDocumentView:` (page 636) (NSClipView)

Related Sample Code
FunHouse
QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit
Rulers
WhackedTV

Declared In
`NSScrollView.h`

setDrawsBackground:

Sets whether the receiver draws its background.

```
- (void)setDrawsBackground:(BOOL)flag
```

Discussion
If *flag* is NO, copy-on-scroll is automatically disabled.

If your NSScrollView encloses an NSClipView sending a `setDrawsBackground:` message with a parameter of NO to the NSScrollView has the added effect of sending the NSClipView a `setCopiesOnScroll:` message with a parameter of NO. The side effect of sending the `setDrawsBackground:` message directly to the NSClipView instead would be the appearance of "trails" (vestiges of previous drawing) in the document view as it is scrolled.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawsBackground` (page 2347)
- `copiesOnScroll` (page 633) (NSClipView)
- `setDrawsBackground:` (page 637) (NSClipView)

Related Sample Code
Quartz Composer QCTV

Declared In
NSScrollView.h

setHasHorizontalRuler:

Determines whether the receiver keeps a horizontal ruler object.

```
- (void)setHasHorizontalRuler:(BOOL)flag
```

Discussion
If *flag* is YES, the receiver allocates a horizontal ruler the first time it's needed. Display of rulers is handled independently with the `setRulersVisible:` (page 2361) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hasHorizontalRuler` (page 2347)
- `horizontalRulerView` (page 2350)
- `setHasVerticalRuler:` (page 2357)

Related Sample Code
Rulers
Sketch+Accessibility
Sketch-112

Declared In
NSScrollView.h

setHasHorizontalScroller:

Determines whether the receiver keeps a horizontal scroller.

```
- (void)setHasHorizontalScroller:(BOOL)flag
```

Discussion
If *flag* is YES, the receiver allocates and displays a horizontal scroller as needed. An NSScrollView by default has neither a horizontal nor a vertical scroller.

Availability
Available in Mac OS X v10.0 and later.

See Also
- hasHorizontalScroller (page 2347)
- horizontalScroller (page 2350)
- setHasVerticalScroller: (page 2358)

Related Sample Code
ColorMatching
FunHouse
Quartz Composer QCTV
Quartz Composer WWDC 2005 TextEdit
WhackedTV

Declared In
NSScrollView.h

setHasVerticalRuler:

Determines whether the receiver keeps a vertical ruler object.

```
- (void)setHasVerticalRuler:(BOOL)flag
```

Discussion
If *flag* is YES, the receiver allocates a vertical ruler the first time it's needed. Display of rulers is handled independently with the setRulersVisible: (page 2361) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- hasVerticalRuler (page 2348)
- verticalRulerView (page 2365)
- setHasHorizontalRuler: (page 2356)
- setRulersVisible: (page 2361)

Related Sample Code
Rulers
Sketch+Accessibility
Sketch-112

Declared In
`NSScrollView.h`

setHasVerticalScroller:

Determines whether the receiver keeps a vertical scroller.

```
- (void)setHasVerticalScroller:(BOOL)flag
```

Discussion
If *flag* is `YES`, the receiver allocates and displays a vertical scroller as needed. An NSScrollView by default has neither a vertical nor a horizontal scroller.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hasVerticalScroller` (page 2348)
- `verticalScroller` (page 2366)
- `setHasHorizontalScroller:` (page 2357)

Related Sample Code
ColorMatching
FancyAbout
FunHouse
Quartz Composer QCTV
WhackedTV

Declared In
`NSScrollView.h`

setHorizontalLineScroll:

Sets the amount by which the receiver scrolls itself horizontally when scrolling line by line to *aFloat*, expressed in the content view's coordinate system.

```
- (void)setHorizontalLineScroll:(CGFloat)aFloat
```

Discussion
This amount is the amount used when the user clicks the scroll arrows on the horizontal scroll bar without holding down a modifier key. When displaying text in an NSScrollView, for example, you might set this amount to the height of a single line of text in the default font.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lineScroll` (page 2350)
- `setPageScroll:` (page 2361)

Declared In
`NSScrollView.h`

setHorizontalPageScroll:

Sets the amount of the document view kept visible when scrolling horizontally page by page to *aFloat*, expressed in the content view's coordinate system.

```
- (void)setHorizontalPageScroll:(CGFloat)aFloat
```

Discussion
This amount is used when the user clicks the scroll arrows on the horizontal scroll bar while holding down the Option key.

This amount expresses the context that remains when the receiver scrolls by one page, allowing the user to orient to the new display. It differs from the line scroll amount, which indicates how far the document view moves. The page scroll amount is the amount common to the content view before and after the document view is scrolled by one page. Thus, setting the page scroll amount to 0.0 implies that the entire visible portion of the document view is replaced when a page scroll occurs.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `pageScroll` (page 2351)
- `setLineScroll:` (page 2360)

Declared In
`NSScrollView.h`

setHorizontalRulerView:

Sets the receiver's horizontal ruler view to *aRulerView*.

```
- (void)setHorizontalRulerView:(NSRulerView *)aRulerView
```

Discussion
You can use this method to override the default ruler class set using the class method `setRulerViewClass:` (page 2343). Display of rulers is controlled using the `setRulersVisible:` (page 2361) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `horizontalRulerView` (page 2350)
- `setHasHorizontalRuler:` (page 2356)
- `setVerticalRulerView:` (page 2363)

- setRulersVisible: (page 2361)

Declared In
NSScrollView.h

setHorizontalScroller:

Sets the receiver's horizontal scroller to *aScroller*, establishing the appropriate target-action relationships between them.

```
- (void)setHorizontalScroller:(NSScroller *)aScroller
```

Discussion
To make sure the scroller is visible, invoke the setHasHorizontalScroller: (page 2357) method with an argument of YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- horizontalScroller (page 2350)
- setVerticalScroller: (page 2364)

Declared In
NSScrollView.h

setLineScroll:

Sets the horizontal and vertical line scroll amounts to *aFloat*.

```
- (void)setLineScroll:(CGFloat)aFloat
```

Discussion
The line scroll is the amount by which the receiver scrolls itself when scrolling line by line, expressed in the content view's coordinate system. It's used when the user clicks the scroll arrows without holding down a modifier key. When displaying text in an NSScrollView, for example, you might set this value to the height of a single line of text in the default font.

As part of its implementation, this method calls setVerticalLineScroll: (page 2362) and setHorizontalLineScroll: (page 2358).

Availability
Available in Mac OS X v10.0 and later.

See Also
- verticalLineScroll (page 2364)
- horizontalLineScroll (page 2349)

Declared In
NSScrollView.h

setPageScroll:

Sets the horizontal and vertical page scroll amounts to *aFloat*.

```
- (void)setPageScroll:(CGFloat)aFloat
```

Discussion
The page scroll is the amount of the document view kept visible when scrolling page by page to *aFloat*, expressed in the content view's coordinate system. It's used when the user clicks the scroll arrows while holding down the Option key.

This amount expresses the context that remains when the receiver scrolls by one page, allowing the user to orient to the new display. It differs from the line scroll amount, which indicates how far the document view moves. The page scroll amount is the amount common to the content view before and after the document view is scrolled by one page. Thus, setting the page scroll amount to 0.0 implies that the entire visible portion of the document view is replaced when a page scroll occurs.

As part of its implementation, this method calls `setVerticalPageScroll:` (page 2363) and `setHorizontalPageScroll:` (page 2359).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `verticalPageScroll` (page 2365)
- `verticalLineScroll` (page 2364)

Declared In
`NSScrollView.h`

setRulersVisible:

Determines whether the receiver displays its rulers.

```
- (void)setRulersVisible:(BOOL)flag
```

Discussion
If *flag* is `YES`, the receiver displays its rulers (creating them if needed). If *flag* is `NO`, the receiver doesn't display its rulers.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rulersVisible` (page 2352)
- `hasHorizontalRuler` (page 2347)
- `hasVerticalRuler` (page 2348)

Related Sample Code
Rulers
Sketch+Accessibility
Sketch-112

Declared In
NSScrollView.h

setScrollsDynamically:

Determines whether the receiver redraws its document view while scrolling continuously.

```
- (void)setScrollsDynamically:(BOOL)flag
```

Discussion
If *flag* is YES it does; if *flag* is NO it redraws only when the scroller knob is released. NSScrollView scrolls dynamically by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- scrollsDynamically (page 2352)

Related Sample Code
WhackedTV

Declared In
NSScrollView.h

setVerticalLineScroll:

Sets the amount by which the receiver scrolls itself vertically when scrolling line by line to *aFloat*, expressed in the content view's coordinate system.

```
- (void)setVerticalLineScroll:(CGFloat)aFloat
```

Discussion
This value is the amount used when the user clicks the scroll arrows on the vertical scroll bar without holding down a modifier key. When displaying text in an NSScrollView, for example, you might set this value to the height of a single line of text in the default font.

Availability
Available in Mac OS X v10.0 and later.

See Also
- verticalLineScroll (page 2364)
- setHorizontalLineScroll: (page 2358)
- lineScroll (page 2350)
- setVerticalPageScroll: (page 2363)

Declared In
NSScrollView.h

setVerticalPageScroll:

Sets the amount of the document view kept visible when scrolling vertically page by page to *aFloat*, expressed in the content view's coordinate system.

```
- (void)setVerticalPageScroll:(CGFloat)aFloat
```

Discussion
This amount is used when the user clicks the scroll arrows on the vertical scroll bar while holding down the Option key.

This amount expresses the context that remains when the receiver scrolls by one page, allowing the user to orient to the new display. It differs from the line scroll amount, which indicates how far the document view moves. The page scroll amount is the amount common to the content view before and after the document view is scrolled by one page. Thus, setting the page scroll amount to 0.0 implies that the entire visible portion of the document view is replaced when a page scroll occurs.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `verticalPageScroll` (page 2365)
- `setHorizontalPageScroll:` (page 2359)
- `pageScroll` (page 2351)
- `setVerticalLineScroll:` (page 2362)

Declared In
`NSScrollView.h`

setVerticalRulerView:

Sets the receiver's vertical ruler view to *aRulerView*.

```
- (void)setVerticalRulerView:(NSRulerView *)aRulerView
```

Discussion
You can use this method to override the default ruler class set using the class method `setRulerViewClass:` (page 2343). Display of rulers is controlled using the `setRulersVisible:` (page 2361) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `verticalRulerView` (page 2365)
- `setHasVerticalRuler:` (page 2357)
- `setHorizontalRulerView:` (page 2359)
- `setRulersVisible:` (page 2361)

Declared In
`NSScrollView.h`

setVerticalScroller:

Sets the receiver's vertical scroller to *aScroller*, establishing the appropriate target-action relationships between them.

```
- (void)setVerticalScroller:(NSScroller *)aScroller
```

Discussion
To make sure the scroller is visible, invoke the `setHasVerticalScroller:` (page 2358) method with an argument of `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `verticalScroller` (page 2366)
- `setHorizontalScroller:` (page 2360)

Declared In
`NSScrollView.h`

tile

Lays out the components of the receiver: the content view, the scrollers, and the ruler views.

```
- (void)tile
```

Discussion
You rarely need to invoke this method, but subclasses may override it to manage additional components.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility

Declared In
`NSScrollView.h`

verticalLineScroll

Returns the amount by which the receiver scrolls itself vertically when scrolling line by line, expressed in the content view's coordinate system.

```
- (CGFloat)verticalLineScroll
```

Discussion
This amount is used when the user clicks the scroll arrows on the vertical scroll bar without holding down a modifier key.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setVerticalLineScroll:` (page 2362)
- `horizontalLineScroll` (page 2349)
- `setLineScroll:` (page 2360)
- `verticalPageScroll` (page 2365)

Declared In
`NSScrollView.h`

verticalPageScroll

Returns the amount of the document view kept visible when scrolling vertically page by page, expressed in the content view's coordinate system.

```
- (CGFloat)verticalPageScroll
```

Discussion
This amount is used when the user clicks the scroll arrows on the vertical scroll bar while holding down the Option key.

This amount expresses the context that remains when the receiver scrolls by one page, allowing the user to orient to the new display. It differs from the line scroll amount, which indicates how far the document view moves. The page scroll amount is the amount common to the content view before and after the document view is scrolled by one page.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setVerticalPageScroll:` (page 2363)
- `horizontalPageScroll` (page 2349)
- `setPageScroll:` (page 2361)
- `verticalLineScroll` (page 2364)

Declared In
`NSScrollView.h`

verticalRulerView

Returns the receiver's vertical ruler view, regardless of whether the receiver is currently displaying it, or `nil` if the receiver has none.

```
- (NSRulerView *)verticalRulerView
```

Discussion
If the receiver is set to display a vertical ruler view and doesn't yet have one, this method creates an instance of the ruler view class set using the class method `setRulerViewClass:` (page 2343). Display of rulers is controlled using the `setRulersVisible:` (page 2361) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hasVerticalRuler` (page 2348)
- `horizontalRulerView` (page 2350)

Related Sample Code
Rulers
Sketch-112

Declared In
`NSScrollView.h`

verticalScroller

Returns the receiver's vertical scroller, regardless of whether the receiver is currently displaying it, or `nil` if the receiver has none.

```
- (NSScroller *)verticalScroller
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hasVerticalScroller` (page 2348)
- `setVerticalScroller:` (page 2364)
- `horizontalScroller` (page 2350)

Related Sample Code
FunHouse
Quartz Composer QCTV
WhackedTV

Declared In
`NSScrollView.h`

NSSearchField Class Reference

Inherits from	NSTextField : NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSTextField) NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSSearchField.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Search Fields
Related sample code	IdentitySample iSpend PDFKitLinker2 SearchField SpotlightAPI

Overview

An `NSSearchField` object implements a text field control that is optimized for performing text-based searches. The control provides a customized text field for entering search data, a search button, a cancel button, and a pop-up icon menu for listing recent search strings and custom search categories.

An `NSSearchField` object wraps an `NSSearchFieldCell` object. Access to most search field attributes occurs through the cell, which provides a more comprehensive programmatic interface for manipulating the search field. You can use an `NSSearchField` object though to manipulate some aspects of the search field. For additional information about search fields and how to manipulate them, see the NSSearchFieldCell class.

Tasks

Managing Recent Searches

`- setRecentSearches:` (page 2369)
Sets the list of recent search strings to list in the pop-up icon menu of the receiver.

- `recentSearches` (page 2368)
  Returns the list of recent search strings for the control.

Managing Autosave Name

- `setRecentsAutosaveName:` (page 2369)
  Sets the autosave name under which the receiver automatically archives the list of recent search strings.
- `recentsAutosaveName` (page 2368)
  Returns the key under which the prior list of recent search strings has been archived.

Instance Methods

recentsAutosaveName

Returns the key under which the prior list of recent search strings has been archived.

```
- (NSString *)recentsAutosaveName
```

Return Value
The autosave name, which is used as a key in the standard user defaults to save the recent searches. The default value is `nil`, which causes searches not to be autosaved.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setRecentsAutosaveName:` (page 2369)

Declared In
`NSSearchField.h`

recentSearches

Returns the list of recent search strings for the control.

```
- (NSArray *)recentSearches
```

Return Value
An array of `NSString` objects, each of which contains a search string either displayed in the search menu or from a recent autosave archive. If there have been no recent searches and no prior searches saved under an autosave name, this array may be empty.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setRecentSearches:` (page 2369)

Declared In
NSSearchField.h

setRecentsAutosaveName:

Sets the autosave name under which the receiver automatically archives the list of recent search strings.

```
- (void)setRecentsAutosaveName:(NSString *)name
```

Parameters

name

The autosave name, which is used as a key in the standard user defaults to save the recent searches. If you specify `nil` or an empty string for this parameter, no autosave name is set and searches are not autosaved.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `recentsAutosaveName` (page 2368)

Declared In
NSSearchField.h

setRecentSearches:

Sets the list of recent search strings to list in the pop-up icon menu of the receiver.

```
- (void)setRecentSearches:(NSArray *)searches
```

Parameters

searches

An array of `NSString` objects containing the search strings.

Discussion
You might use this method to set the recent list of searches from an archived copy.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `recentSearches` (page 2368)

Declared In
NSSearchField.h

NSSearchFieldCell Class Reference

Inherits from	NSTextFieldCell : NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSSearchFieldCell.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Search Fields

Overview

The `NSSearchFieldCell` class defines the programmatic interface for text fields that are optimized for text-based searches. An `NSSearchFieldCell` object is "wrapped" by an NSSearchField control object, which directly inherits from the `NSTextField` class. The search field implemented by these classes presents a standard user interface for searches, including a search button, a cancel button, and a pop-up icon menu for listing recent search strings and custom search categories.

When the user types and then pauses, the cell's action message is sent to its target. You can query the cell's string value for the current text to search for. Do not rely on the sender of the action to be an `NSMenu` object because the menu may change. If you need to change the menu, modify the search menu template and call the `setSearchMenuTemplate:` method to update.

Tasks

Managing Buttons

- `setSearchButtonCell:` (page 2380)
  Sets the button cell used to display the search-button image
- `searchButtonCell` (page 2376)
  Returns the button cell used to display the search-button image.
- `resetSearchButtonCell` (page 2375)
  Resets the search button cell to its default attributes.

- `setCancelButtonCell:` (page 2378)
  Sets the button cell object used to display the cancel-button image
- `cancelButtonCell` (page 2373)
  Returns the button cell object used to display the cancel-button image.
- `resetCancelButtonCell` (page 2375)
  Resets the cancel button cell to its default attributes.

Custom Layout

- `searchTextRectForBounds:` (page 2377)
  Modifies the bounding rectangle for the search-text field cell.
- `searchButtonRectForBounds:` (page 2376)
  Modifies the bounding rectangle for the search button cell.
- `cancelButtonRectForBounds:` (page 2373)
  Modifies the bounding rectangle for the cancel button cell.

Managing Menu Templates

- `setSearchMenuTemplate:` (page 2380)
  Sets the menu template object used to dynamically construct the receiver's pop-up icon menu.
- `searchMenuTemplate` (page 2377)
  Returns the menu template object used to dynamically construct the search pop-up icon menu.

Managing Search Modes

- `setSendsWholeSearchString:` (page 2381)
  Sets whether the receiver sends the search action message when the user clicks the search button (or presses return) or after each keystroke.
- `sendsWholeSearchString` (page 2378)
  Returns a Boolean value indicating whether the receiver sends the search action message when the user clicks the search button (or presses return) or after each keystroke.
- `sendsSearchStringImmediately` (page 2378)
  Returns a Boolean value indicating whether the receiver sends its action immediately upon being notified of changes to the search field text or after a brief pause.
- `setSendsSearchStringImmediately:` (page 2381)
  Sets whether the cell sends its action message to the target immediately upon notification of any changes to the search field text or after a brief pause.

Managing Recent Search Strings

- `setMaximumRecents:` (page 2379)
  Sets the maximum number of search strings that can appear in the search menu

- `maximumRecents` (page 2374)
  Returns the maximum number of recent search strings to display in the custom search menu.
- `setRecentSearches:` (page 2380)
  Sets the list of recent search strings to list in the pop-up icon menu of the receiver.
- `recentSearches` (page 2375)
  Returns the list of recent search strings for the control.
- `setRecentsAutosaveName:` (page 2379)
  Sets the autosave name under which the receiver automatically archives the list of recent search strings.
- `recentsAutosaveName` (page 2374)
  Returns the key under which the prior list of recent search strings has been archived.

Instance Methods

cancelButtonCell

Returns the button cell object used to display the cancel-button image.

```
- (NSButtonCell *)cancelButtonCell
```

Return Value
The cancel button cell.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setCancelButtonCell:` (page 2378)
- `resetCancelButtonCell` (page 2375)

Declared In
`NSSearchFieldCell.h`

cancelButtonRectForBounds:

Modifies the bounding rectangle for the cancel button cell.

```
- (NSRect)cancelButtonRectForBounds:(NSRect)rect
```

Parameters
rect
  The current bounding rectangle for the cancel button.

Return Value
The updated bounding rectangle to use for the cancel button. The default value is the value passed into the *rect* parameter.

Discussion
Subclasses can override this method to return a new bounding rectangle for the cancel button cell. You might use this method to provide a custom layout for the search field control.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `searchButtonRectForBounds:` (page 2376)
- `searchTextRectForBounds:` (page 2377)

Declared In
`NSSearchFieldCell.h`

maximumRecents

Returns the maximum number of recent search strings to display in the custom search menu.

```
- (NSInteger)maximumRecents
```

Return Value
The maximum number of search strings that can appear in the menu. This value is between `0` and `254`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setMaximumRecents:` (page 2379)

Declared In
`NSSearchFieldCell.h`

recentsAutosaveName

Returns the key under which the prior list of recent search strings has been archived.

```
- (NSString *)recentsAutosaveName
```

Return Value
The autosave name, which is used as a key in the standard user defaults to save the recent searches. The default value is `nil`, which causes searches not to be autosaved.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setRecentsAutosaveName:` (page 2379)

Declared In
`NSSearchFieldCell.h`

recentSearches

Returns the list of recent search strings for the control.

```
- (NSArray *)recentSearches
```

Return Value
An array of `NSString` objects, each of which contains a search string either displayed in the search menu or from a recent autosave archive. If there have been no recent searches and no prior searches saved under an autosave name, this array may be empty.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setRecentsAutosaveName:` (page 2379)
- `setRecentSearches:` (page 2380)

Declared In
`NSSearchFieldCell.h`

resetCancelButtonCell

Resets the cancel button cell to its default attributes.

```
- (void)resetCancelButtonCell
```

Discussion
This method resets the target, action, regular image, and pressed image. By default, when users click the cancel button, the `delete:` action message is sent up the responder chain to the first `NSText` object that can handle it. This method gives you a way to customize the cancel button for specific situations and then reset the button defaults without having to undo changes individually.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setCancelButtonCell:` (page 2378)
- `cancelButtonCell` (page 2373)

Declared In
`NSSearchFieldCell.h`

resetSearchButtonCell

Resets the search button cell to its default attributes.

```
- (void)resetSearchButtonCell
```

Discussion
This method resets the target, action, regular image, and pressed image. By default, when users click the search button or press the Return key, the action defined for the receiver is sent to its designated target. This method gives you a way to customize the search button for specific situations and then reset the button defaults without having to undo changes individually.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setSearchButtonCell:` (page 2380)
- `searchButtonCell` (page 2376)

Declared In
`NSSearchFieldCell.h`

searchButtonCell

Returns the button cell used to display the search-button image.

```
- (NSButtonCell *)searchButtonCell
```

Return Value
The search button cell.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setSearchButtonCell:` (page 2380)
- `resetSearchButtonCell` (page 2375)

Declared In
`NSSearchFieldCell.h`

searchButtonRectForBounds:

Modifies the bounding rectangle for the search button cell.

```
- (NSRect)searchButtonRectForBounds:(NSRect)rect
```

Parameters
rect
The current bounding rectangle for the search button.

Return Value
The updated bounding rectangle to use for the search button. The default value is the value passed into the *rect* parameter.

Discussion
Subclasses can override this method to return a new bounding rectangle for the search button cell. You might use this method to provide a custom layout for the search field control.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `cancelButtonRectForBounds:` (page 2373)
- `searchTextRectForBounds:` (page 2377)

Declared In
`NSSearchFieldCell.h`

searchMenuTemplate

Returns the menu template object used to dynamically construct the search pop-up icon menu.

```
- (NSMenu *)searchMenuTemplate
```

Return Value
The current menu template.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setSearchMenuTemplate:` (page 2380)

Declared In
`NSSearchFieldCell.h`

searchTextRectForBounds:

Modifies the bounding rectangle for the search-text field cell.

```
- (NSRect)searchTextRectForBounds:(NSRect)rect
```

Parameters
rect
The current bounding rectangle for the search text field.

Return Value
The updated bounding rectangle to use for the search text field. The default value is the value passed into the *rect* parameter.

Discussion
Subclasses can override this method to return a new bounding rectangle for the text-field cell object. You might use this method to provide a custom layout for the search field control.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `cancelButtonRectForBounds:` (page 2373)
- `searchButtonRectForBounds:` (page 2376)

Declared In
NSSearchFieldCell.h

sendsSearchStringImmediately

Returns a Boolean value indicating whether the receiver sends its action immediately upon being notified of changes to the search field text or after a brief pause.

```
- (BOOL)sendsSearchStringImmediately
```

Return Value
YES if the cell sends its action immediately upon notification of any changes to the search field; otherwise, NO.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setSendsSearchStringImmediately: (page 2381)

Declared In
NSSearchFieldCell.h

sendsWholeSearchString

Returns a Boolean value indicating whether the receiver sends the search action message when the user clicks the search button (or presses return) or after each keystroke.

```
- (BOOL)sendsWholeSearchString
```

Return Value
YES if the action message is sent all at once when the user clicks the search button or presses return; otherwise, NO if the search string is sent after each keystroke. The default value is NO.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setSendsWholeSearchString: (page 2381)

Declared In
NSSearchFieldCell.h

setCancelButtonCell:

Sets the button cell object used to display the cancel-button image

```
- (void)setCancelButtonCell:(NSButtonCell *)cell
```

Parameters
cell
The cancel button cell.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `cancelButtonCell` (page 2373)
- `resetCancelButtonCell` (page 2375)

Declared In
`NSSearchFieldCell.h`

setMaximumRecents:

Sets the maximum number of search strings that can appear in the search menu

```
- (void)setMaximumRecents:(NSInteger)maxRecents
```

Parameters

maxRecents

The maximum number of search strings that can appear in the menu. This value can be between `0` and `254`. Specifying a value less than `0` sets the value to the default, which is `10`. Specifying a value greater than `254` sets the maximum to `254`.

Discussion
When the limit is exceeded, the oldest search string on the menu is dropped.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `maximumRecents` (page 2374)

Declared In
`NSSearchFieldCell.h`

setRecentsAutosaveName:

Sets the autosave name under which the receiver automatically archives the list of recent search strings.

```
- (void)setRecentsAutosaveName:(NSString *)name
```

Parameters

name

The autosave name, which is used as a key in the standard user defaults to save the recent searches. If you specify `nil` or an empty string for this parameter, no autosave name is set and searches are not autosaved.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `recentsAutosaveName` (page 2374)
- `setRecentSearches:` (page 2380)

Declared In
NSSearchFieldCell.h

setRecentSearches:

Sets the list of recent search strings to list in the pop-up icon menu of the receiver.

```
- (void)setRecentSearches:(NSArray *)searches
```

Parameters

searches

An array of `NSString` objects containing the search strings.

Discussion
You might use this method to set the recent list of searches from an archived copy.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `recentSearches` (page 2375)

Declared In
NSSearchFieldCell.h

setSearchButtonCell:

Sets the button cell used to display the search-button image

```
- (void)setSearchButtonCell:(NSButtonCell *)cell
```

Parameters

cell

The search button cell.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `searchButtonCell` (page 2376)
- `resetSearchButtonCell` (page 2375)

Declared In
NSSearchFieldCell.h

setSearchMenuTemplate:

Sets the menu template object used to dynamically construct the receiver's pop-up icon menu.

```
- (void)setSearchMenuTemplate:(NSMenu *)menu
```

Parameters

menu

The menu template to use.

Discussion

The receiver looks for the tag constants described in "Menu tags" (page 2382) to determine how to populate the menu with items related to recent searches. See "Configuring a Search Menu" for a sample of how you might set up the search menu template.

Availability

Available in Mac OS X v10.3 and later.

See Also

- searchMenuTemplate (page 2377)

Declared In

NSSearchFieldCell.h

setSendsSearchStringImmediately:

Sets whether the cell sends its action message to the target immediately upon notification of any changes to the search field text or after a brief pause.

```
- (void)setSendsSearchStringImmediately:(BOOL)flag
```

Parameters

flag

YES to send the cell's action immediately upon notification of any changes to the search field; otherwise, NO if you want the cell to pause briefly before sending its action message. Pausing gives the user the opportunity to type more text into the search field before initiating the search.

Availability

Available in Mac OS X v10.4 and later.

See Also

- sendsSearchStringImmediately (page 2378)

Declared In

NSSearchFieldCell.h

setSendsWholeSearchString:

Sets whether the receiver sends the search action message when the user clicks the search button (or presses return) or after each keystroke.

```
- (void)setSendsWholeSearchString:(BOOL)flag
```

Parameters

flag

YES to send the action message all at once when the user clicks the search button or presses return; otherwise, NO to send the search string after each keystroke.

Availability

Available in Mac OS X v10.3 and later.

See Also
- sendsWholeSearchString (page 2378)

Declared In
NSSearchFieldCell.h

Constants

Menu tags

The NSSearchFieldCell class uses these tag constants for identifying special menu items in the search-menu template set by setSearchMenuTemplate: (page 2380). When an NSSearchFieldCell object dynamically constructs the actual search menu from this template, it shows or hides the tagged items as directed.

```
#define NSSearchFieldRecentsTitleMenuItemTag     1000
#define NSSearchFieldRecentsMenuItemTag          1001
#define NSSearchFieldClearRecentsMenuItemTag     1002
#define NSSearchFieldNoRecentsMenuItemTag        1003
```

Constants

NSSearchFieldRecentsTitleMenuItemTag
: Identifies the menu item that is the title of the menu group for recent search strings.

  This item is hidden if there are no recent strings.

  You may use this tagged item for separator characters that also do not appear if there are no recent strings to display.

  Available in Mac OS X v10.3 and later.

  Declared in NSSearchFieldCell.h.

NSSearchFieldRecentsMenuItemTag
: Identifies where recent search strings should appear in the “recents” menu group.

  Available in Mac OS X v10.3 and later.

  Declared in NSSearchFieldCell.h.

NSSearchFieldClearRecentsMenuItemTag
: Identifies the menu item for clearing the current set of recent string searches in the menu.

  This item is hidden if there are no recent strings.

  Available in Mac OS X v10.3 and later.

  Declared in NSSearchFieldCell.h.

NSSearchFieldNoRecentsMenuItemTag
: Identifies the menu item that describes a lack of recent search strings (for example, “No recent searches”).

  This item is hidden if there have been recent searches.

  Available in Mac OS X v10.3 and later.

  Declared in NSSearchFieldCell.h.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSSearchFieldCell.h`

NSSecureTextField Class Reference

Inherits from	NSTextField : NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSTextField) NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSSecureTextField.h
Companion guide	Text Fields
Related sample code	IdentitySample NameAndPassword

Overview

`NSSecureTextField` is a subclass of `NSTextField` that hides its text from display or other access via the user interface. It's suitable for use as a password-entry object or for any item in which a secure value must be kept.

`NSSecureTextField` uses NSSecureTextFieldCell to implement its user interface.

NSSecureTextFieldCell Class Reference

Inherits from	NSTextFieldCell : NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSSecureTextField.h
Companion guide	Text Fields

Overview

`NSSecureTextFieldCell` works with NSSecureTextField to provide a text field whose value is guarded from user examination. It overrides the general cell use of the field editor to provide its own field editor, which doesn't display text or allow the user to cut or copy its value.

Tasks

Working with Character Echo

`- echosBullets` (page 2387)
Returns whether the receiver echoes a bullet character rather than each character typed.

`- setEchosBullets:` (page 2388)
Sets whether the receiver echoes bullets for each character typed.

Instance Methods

echosBullets

Returns whether the receiver echoes a bullet character rather than each character typed.

```
- (BOOL)echosBullets
```

Discussion
Default is `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setEchosBullets:` (page 2388)

Declared In
`NSSecureTextField.h`

setEchosBullets:

Sets whether the receiver echoes bullets for each character typed.

```
- (void)setEchosBullets:(BOOL)flag
```

Discussion
If `YES`, bullets are echoed. If `NO`, the cursor is moved for each character typed, but nothing is displayed.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- echosBullets` (page 2387)

Declared In
`NSSecureTextField.h`

NSSegmentedCell Class Reference

Inherits from	NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSSegmentedCell.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Segmented Control Programming Guide

Overview

The `NSSegmentedCell` class implements the appearance and behavior of a horizontal button divided into multiple segments. This class is used in conjunction with the `NSSegmentedControl` class to implement a segmented control.

You can customize the attributes of a segmented control using the methods of `NSSegmentedCell`. To customize the appearance of individual segments, you can also subclass and override the `drawSegment:inFrame:withView:` (page 2391) method.

Tasks

Specifying the Number of Segments

- `setSegmentCount:` (page 2399)
  Sets the number of segments in the receiver.
- `segmentCount` (page 2396)
  Returns the number of segments in the receiver.

Specifying the Selected Segment

- `setSelectedSegment:` (page 2401)
  Sets the selected segment of the receiver.

- `setSelected:forSegment:` (page 2400)
  Sets the selection state of the specified segment.
- `selectSegmentWithTag:` (page 2397)
  Selects the segment with the specified tag.
- `makeNextSegmentKey` (page 2394)
  Selects the next segment.
- `makePreviousSegmentKey` (page 2395)
  Selects the previous segment.
- `selectedSegment` (page 2396)
  Returns the index of the selected segment of the receiver.
- `isSelectedForSegment:` (page 2394)
  Returns a Boolean value indicating whether the specified segment is selected,

Specifying the Tracking Mode

- `setTrackingMode:` (page 2402)
  Sets the tracking mode for the receiver.
- `trackingMode` (page 2404)
  Returns the tracking mode of the receiver.

Configuring Individual Segments

- `setLabel:forSegment:` (page 2398)
  Sets the label for the specified segment.
- `labelForSegment:` (page 2394)
  Returns the label of the specified segment.
- `setImage:forSegment:` (page 2397)
  Sets the image for the specified segment.
- `imageForSegment:` (page 2392)
  Returns the image associated with the specified segment.
- `setImageScaling:forSegment:` (page 2398)
  Sets the image scaling mode for the specified segment.
- `imageScalingForSegment:` (page 2392)
  Returns the image scaling mode associated with the specified segment.
- `setWidth:forSegment:` (page 2402)
  Sets the width of the specified segment.
- `widthForSegment:` (page 2404)
  Returns the width of the specified segment.
- `setEnabled:forSegment:` (page 2397)
  Sets the enabled state of the specified segment
- `isEnabledForSegment:` (page 2393)
  Returns a Boolean value indicating whether the specified segment is enabled.

- `setMenu:forSegment:` (page 2399)
  Sets the menu for the specified segment.
- `menuForSegment:` (page 2395)
  Returns the menu for the specified segment.
- `setToolTip:forSegment:` (page 2402)
  Sets the tool tip for the specified segment.
- `toolTipForSegment:` (page 2403)
  Returns the tool tip of the specified segment.
- `setTag:forSegment:` (page 2401)
  Sets the tag for the specified segment.
- `tagForSegment:` (page 2403)
  Returns the tag of the specified segment.

Drawing Custom Content

- `drawSegment:inFrame:withView:` (page 2391)
  Draws the segment in the specified view.

Specifying Segment Visual Styles

- `interiorBackgroundStyleForSegment:` (page 2393)
  Returns the interior background style for the specified segment.
- `segmentStyle` (page 2396)
  Returns the visual style used to display the receiver.
- `setSegmentStyle:` (page 2400)
  Sets the visual style used to display the receiver.

Instance Methods

drawSegment:inFrame:withView:

Draws the segment in the specified view.

```
- (void)drawSegment:(NSInteger)segment inFrame:(NSRect)frame withView:(NSView
    *)controlView
```

Parameters

segment
The index of the segment to draw. This method raises an `NSRangeException` if the index is out of bounds.

frame
The rectangle in which to draw the segment. This rectangle is specified in user space coordinates of the specified view.

controlView
The view in which to draw the segment.

Discussion
You can override this method to provide a custom appearance for segmented controls. You should not call this method directly. It is called for you automatically by the control when it needs to be redrawn.

Availability
Available in Mac OS X v10.3 and later.

See Also
- drawWithFrame:inView: (page 554) (NSCell)

Declared In
NSSegmentedCell.h

imageForSegment:

Returns the image associated with the specified segment.

```
- (NSImage *)imageForSegment:(NSInteger)segment
```

Parameters
segment
The index of the segment whose image you want to get. This method raises an NSRangeException if the index is out of bounds.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setImage:forSegment: (page 2397)

Declared In
NSSegmentedCell.h

imageScalingForSegment:

Returns the image scaling mode associated with the specified segment.

```
- (NSImageScaling)imageScalingForSegment:(NSInteger)segment
```

Parameters
segment
The index of the segment whose image scaling mode you want to get. This method raises an NSRangeException if the index is out of bounds.

Return Value
The scaling mode in use for the specified segment. For the possible values see Segmented Control Visual Styles (page 2419). If no value has been explicitly set NSImageScaleProportionallyDown (page 617) is returned.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setImageScaling:forSegment:` (page 2398)

Declared In
`NSSegmentedCell.h`

interiorBackgroundStyleForSegment:

Returns the interior background style for the specified segment.

```
- (NSBackgroundStyle)interiorBackgroundStyleForSegment:(NSInteger)segment
```

Parameters

segment

The index of the segment whose background style you want to get. This method raises an `NSRangeException` if the index is out of bounds..

Return Value
The background style to use for specified segment See Background Styles in NSCell for possible values.

Discussion
The interior background style describes the surface drawn onto in drawInteriorWithFrame:inView:.

This is both an override point and a useful method to call. In a custom segment cell with a custom bezel you can override this method to describe the surface on a per-segment basis.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSSegmentedCell.h`

isEnabledForSegment:

Returns a Boolean value indicating whether the specified segment is enabled.

```
- (BOOL)isEnabledForSegment:(NSInteger)segment
```

Parameters

segment

The index of the segment whose enabled state you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Return Value
`YES` if the segment is enabled; otherwise, `NO`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setEnabled:forSegment:` (page 2397)

Declared In
`NSSegmentedCell.h`

isSelectedForSegment:

Returns a Boolean value indicating whether the specified segment is selected,

```
- (BOOL)isSelectedForSegment:(NSInteger)segment
```

Parameters

segment

The index of the segment whose selection state you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Return Value

`YES` if the segment is selected; otherwise, `NO`.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- setSelected:forSegment:` (page 2400)

Declared In

`NSSegmentedCell.h`

labelForSegment:

Returns the label of the specified segment.

```
- (NSString *)labelForSegment:(NSInteger)segment
```

Parameters

segment

The index of the segment whose label you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- setLabel:forSegment:` (page 2398)

Declared In

`NSSegmentedCell.h`

makeNextSegmentKey

Selects the next segment.

```
- (void)makeNextSegmentKey
```

Discussion

The next segment is the one to the right of the currently selected segment. For the last segment, the selection wraps back to the beginning of the control.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `makePreviousSegmentKey` (page 2395)

Declared In
`NSSegmentedCell.h`

makePreviousSegmentKey

Selects the previous segment.

```
- (void)makePreviousSegmentKey
```

Discussion
The previous segment is the one to the left of the currently selected segment. For the first segment, the selection wraps around to the last segment of the control.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `makeNextSegmentKey` (page 2394)

Declared In
`NSSegmentedCell.h`

menuForSegment:

Returns the menu for the specified segment.

```
- (NSMenu *)menuForSegment:(NSInteger)segment
```

Parameters
segment
The index of the segment whose menu you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Return Value
The menu associated with the segment; otherwise, `nil`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setMenu:forSegment:` (page 2399)

Declared In
`NSSegmentedCell.h`

segmentCount

Returns the number of segments in the receiver.

```
- (NSInteger)segmentCount
```

Return Value
The number of segments in the receiver.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setSegmentCount:` (page 2399)

Declared In
`NSSegmentedCell.h`

segmentStyle

Returns the visual style used to display the receiver.

```
- (NSSegmentStyle)segmentStyle
```

Return Value
An `NSSegmentStyle` value that specifies the visual display used by the receiver. For possible values see "Segmented Control Visual Styles" (page 2419) in *NSSegmentedControl Class Reference*.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSSegmentedCell.h`

selectedSegment

Returns the index of the selected segment of the receiver.

```
- (NSInteger)selectedSegment
```

Return Value
The index of the currently selected segment, or -1 if no segment is selected. If the receiver allows multiple selections, this method returns the most recently selected segment.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setSelectedSegment:` (page 2401)

Declared In
`NSSegmentedCell.h`

selectSegmentWithTag:

Selects the segment with the specified tag.

```
- (BOOL)selectSegmentWithTag:(NSInteger)tag
```

Parameters

tag

The tag associated with the desired segment.

Return Value

YES if the segment was selected successfully; otherwise, NO.

Discussion

Typically, you use Interface Builder to specify the tag for each segment. You may also set this value programmatically using the setTag:forSegment: (page 2401) method.

Availability

Available in Mac OS X v10.4 and later.

See Also

- setTag:forSegment: (page 2401)

Declared In

NSSegmentedCell.h

setEnabled:forSegment:

Sets the enabled state of the specified segment

```
- (void)setEnabled:(BOOL)flag forSegment:(NSInteger)segment
```

Parameters

flag

YES to enable the segment; otherwise, NO to disable it.

segment

The index of the segment you want to enable or disable. This method raises an NSRangeException if the index is out of bounds.

Availability

Available in Mac OS X v10.3 and later.

See Also

- isEnabledForSegment: (page 2393)

Declared In

NSSegmentedCell.h

setImage:forSegment:

Sets the image for the specified segment.

```
- (void)setImage:(NSImage *)image forSegment:(NSInteger)segment
```

Parameters

image

The image to apply to the segment or `nil` if you want to clear the existing image. Images are not scaled to fit inside a segment. If the image is larger than the available space, it is clipped.

segment

The index of the segment whose image you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- imageForSegment:` (page 2392)

Declared In
`NSSegmentedCell.h`

setImageScaling:forSegment:

Sets the image scaling mode for the specified segment.

```
- (void)setImageScaling:(NSImageScaling)scaling forSegment:(NSInteger)segment
```

Parameters

scaling

The scaling mode to assign to the specified segment. For the possible values see `Segmented Control Visual Styles` (page 2419).

segment

The index of the segment whose image scaling mode you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Discussion
The image scaling mode for a segment affects how the image inside the corresponding cell is positioned and resized when the cell itself grows or shrinks. The image scaling mode does not itself cause the cell to change size in any way. If a cell does not contain an image, the scaling mode has no effect.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- imageScalingForSegment:` (page 2392)

Declared In
`NSSegmentedCell.h`

setLabel:forSegment:

Sets the label for the specified segment.

```
- (void)setLabel:(NSString *)label forSegment:(NSInteger)segment
```

Parameters

label

The label you want to display in the segment. If the width of the string is greater than the width of the segment, the string's text is truncated during drawing.

segment

The index of the segment whose label you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- labelForSegment:` (page 2394)

Declared In
`NSSegmentedCell.h`

setMenu:forSegment:

Sets the menu for the specified segment.

```
- (void)setMenu:(NSMenu *)menu forSegment:(NSInteger)segment
```

Parameters

menu

The menu you want to add to the segment or `nil` to clear the current menu. This menu is displayed when the user clicks and holds the mouse button while the mouse is over the segment.

segment

The index of the segment whose menu you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Discussion
Adding a menu to a segment allows that segment to be used as a pop-up button.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- menuForSegment:` (page 2395)

Declared In
`NSSegmentedCell.h`

setSegmentCount:

Sets the number of segments in the receiver.

```
- (void)setSegmentCount:(NSInteger)count
```

Parameters

count

The number of segments the receiver should have. If this value is less than the number of segments currently in the receiver, segments are removed from the right of the control. Similarly, if the number is greater than the current number of segments, the new segments are added on the right. This value must be between `0` and `2049`.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- segmentCount` (page 2396)

Declared In
`NSSegmentedCell.h`

setSegmentStyle:

Sets the visual style used to display the receiver.

```
- (void)setSegmentStyle:(NSSegmentStyle)segmentStyle
```

Parameters

segmentStyle

An `NSSegmentStyle` value that specifies the visual display used by the receiver. For possible values see "Segmented Control Visual Styles" (page 2419) in *NSSegmentedControl Class Reference*.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSSegmentedCell.h`

setSelected:forSegment:

Sets the selection state of the specified segment.

```
- (void)setSelected:(BOOL)flag forSegment:(NSInteger)segment
```

Parameters

flag

`YES` if you want to select the segment; otherwise, `NO`.

segment

The index of the segment whose selection state you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Discussion
If the receiver allows only a single selection, this method deselects any other selected segments.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `isSelectedForSegment:` (page 2394)

Declared In
`NSSegmentedCell.h`

setSelectedSegment:

Sets the selected segment of the receiver.

```
- (void)setSelectedSegment:(NSInteger)selectedSegment
```

Parameters

selectedSegment

The zero-based index of the desired segment. This method raises an `NSRangeException` if the index is out of bounds.

Discussion
If the receiver allows multiple selections, this method selects the specified segment using `setSelected:forSegment:` (page 2400).

Availability
Available in Mac OS X v10.3 and later.

See Also
- `selectedSegment` (page 2396)

Declared In
`NSSegmentedCell.h`

setTag:forSegment:

Sets the tag for the specified segment.

```
- (void)setTag:(NSInteger)tag forSegment:(NSInteger)segment
```

Parameters

tag

The tag of the segment.

segment

The index of the segment whose tool tag you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `tagForSegment:` (page 2403)

Declared In
`NSSegmentedCell.h`

setToolTip:forSegment:

Sets the tool tip for the specified segment.

```
- (void)setToolTip:(NSString *)toolTip forSegment:(NSInteger)segment
```

Parameters

toolTip

The text of the tool tip you want to display for the segment.

segment

The index of the segment whose tool tip you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Discussion

Tool tips are currently not displayed.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- toolTipForSegment:` (page 2403)

Declared In

`NSSegmentedCell.h`

setTrackingMode:

Sets the tracking mode for the receiver.

```
- (void)setTrackingMode:(NSSegmentSwitchTracking)trackingMode
```

Parameters

trackingMode

The tracking mode to use for the segments. Possible values for *trackingMode* are described in `NSSegmentSwitchTracking` (page 2405).

Availability

Available in Mac OS X v10.3 and later.

See Also

`- trackingMode` (page 2404)

Declared In

`NSSegmentedCell.h`

setWidth:forSegment:

Sets the width of the specified segment.

```
- (void)setWidth:(CGFloat)width forSegment:(NSInteger)segment
```

Parameters

width

The width of the segment, measured in points. Specify the value 0 if you want the segment to be sized to fit the available space automatically.

segment

The index of the segment whose width you want to set. This method raises an NSRangeException if the index is out of bounds.

Availability
Available in Mac OS X v10.3 and later.

See Also
- widthForSegment: (page 2404)

Declared In
NSSegmentedCell.h

tagForSegment:

Returns the tag of the specified segment.

```
- (NSInteger)tagForSegment:(NSInteger)segment
```

Parameters

segment

The index of the segment whose tool tag you want to get. This method raises an NSRangeException if the index is out of bounds.

Return Value
The tag of the segment.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setTag:forSegment: (page 2401)

Declared In
NSSegmentedCell.h

toolTipForSegment:

Returns the tool tip of the specified segment.

```
- (NSString *)toolTipForSegment:(NSInteger)segment
```

Parameters

segment

The index of the segment whose tool tip you want to get. This method raises an NSRangeException if the index is out of bounds.

Return Value
The text of the tool tip.

Discussion
Tool tips are currently not displayed.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setToolTip:forSegment:` (page 2402)

Declared In
`NSSegmentedCell.h`

trackingMode

Returns the tracking mode of the receiver.

```
- (NSSegmentSwitchTracking)trackingMode
```

Return Value
The tracking mode used for the segments Possible values for *trackingMode* are described in `NSSegmentSwitchTracking` (page 2405).The default value is `NSSegmentSwitchTrackingSelectOne` (page 2405).

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setTrackingMode:` (page 2402)

Declared In
`NSSegmentedCell.h`

widthForSegment:

Returns the width of the specified segment.

```
- (CGFloat)widthForSegment:(NSInteger)segment
```

Parameters
segment
The index of the segment whose width you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setWidth:forSegment:` (page 2402)

Declared In
`NSSegmentedCell.h`

Constants

NSSegmentSwitchTracking

An `NSSegmentedCell` object uses the following constants, which describe the various tracking modes for a cell. You access these values using `setTrackingMode:` (page 2402) and `trackingMode` (page 2404) methods of the class.

```
typedef enum {
    NSSegmentSwitchTrackingSelectOne = 0,
    NSSegmentSwitchTrackingSelectAny = 1,
    NSSegmentSwitchTrackingMomentary = 2
} NSSegmentSwitchTracking;
```

Constants

`NSSegmentSwitchTrackingSelectOne`

Only one segment may be selected.

Available in Mac OS X v10.3 and later.

Declared in `NSSegmentedCell.h`.

`NSSegmentSwitchTrackingSelectAny`

Any segment can be selected.

Available in Mac OS X v10.3 and later.

Declared in `NSSegmentedCell.h`.

`NSSegmentSwitchTrackingMomentary`

A segment is selected only when tracking.

Available in Mac OS X v10.3 and later.

Declared in `NSSegmentedCell.h`.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`NSSegmentedCell.h`

NSSegmentedControl Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSSegmentedControl.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Segmented Control Programming Guide
Related sample code	ButtonMadness ImageClient ImageMapExample PDF Annotation Editor PDFKitLinker2

Overview

An `NSSegmentedControl` object implements a horizontal button made of multiple segments.

The `NSSegmentedControl` class uses an `NSSegmentedCell` class to implement much of the control's functionality. Most methods in `NSSegmentedControl` are simply "cover methods" that call the corresponding method in `NSSegmentedCell`. The methods of `NSSegmentedCell` that do not have covers relate to accessing and setting values for tags and tool tips; programatically setting the key segment; and establishing the mode of the control.

The features of a segmented control include:

- Each segment can have an image, text (label), menu, tooltip, and tag
- Either the whole control or individual segments can be enabled or disabled
- There are three tracking modes for segments: select one mode (also known as radio button mode and illustrated by Finder's view mode selection control), momentary mode (as illustrated by Safari's toolbar buttons), or select any mode (where any combination of buttons may be on or off)
- Each segment can be either a fixed width or autosized to fit the contents
- If a segment has text and is marked as autosizing, then the text may be truncated so that the control completely fits

- If an image is too large to fit in a segment, it is clipped
- Full keyboard control of the user interface

Tasks

Specifying Number of Segments

- `setSegmentCount:` (page 2416)
  Sets the number of segments in the receiver.
- `segmentCount` (page 2412)
  Returns the number of segments in the receiver.

Specifying Selected Segment

- `setSelectedSegment:` (page 2418)
  Sets the selected segment of the receiver.
- `selectedSegment` (page 2412)
  Returns the index of the selected segment of the receiver.
- `selectSegmentWithTag:` (page 2413)
  Selects the segment with the specified tag.

Working with Individual Segments

- `setWidth:forSegment:` (page 2418)
  Sets the width of the specified segment.
- `widthForSegment:` (page 2418)
  Returns the width of the specified segment.
- `setImage:forSegment:` (page 2414)
  Sets the image for the specified segment.
- `imageForSegment:` (page 2409)
  Returns the image associated with the specified segment.
- `setLabel:forSegment:` (page 2415)
  Sets the label for the specified segment.
- `labelForSegment:` (page 2411)
  Returns the label of the specified segment
- `setMenu:forSegment:` (page 2415)
  Sets the menu for the specified segment.
- `menuForSegment:` (page 2411)
  Returns the menu for the specified segment
- `setSelected:forSegment:` (page 2417)
  Sets the selection state of the specified segment.

- `isSelectedForSegment:` (page 2410)
  Returns a Boolean value indicating whether the specified segment is selected.
- `setEnabled:forSegment:` (page 2413)
  Sets the enabled state of the specified segment
- `isEnabledForSegment:` (page 2410)
  Returns a Boolean value indicating whether the specified segment is enabled.

Specifying Segment Display

- `setSegmentStyle:` (page 2416)
  Sets the visual style used to display the receiver.
- `segmentStyle` (page 2412)
  Returns the visual style used to display the receiver.
- `setImageScaling:forSegment:` (page 2414)
  Sets the scaling mode used to display the specified segment's image.
- `imageScalingForSegment:` (page 2409)
  Returns the scaling mode used to display the specified segment's image.

Instance Methods

imageForSegment:

Returns the image associated with the specified segment.

```
- (NSImage *)imageForSegment:(NSInteger)segment
```

Parameters

segment
The index of the segment whose image you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Return Value
The image associated with the segment; otherwise, `nil`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setImage:forSegment:` (page 2414)

Declared In
`NSSegmentedControl.h`

imageScalingForSegment:

Returns the scaling mode used to display the specified segment's image.

```
- (NSImageScaling)imageScalingForSegment:(NSInteger)segment
```

Parameters

segment

The index of the segment whose enabled state you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Return Value

One of the image scaling constants. For a list of possible values, see `NSImageScaling` (page 617). The default value is `NSImageScaleProportionallyDown` (page 617).

Availability

Available in Mac OS X v10.5 and later.

See Also

`- setImageScaling:forSegment:` (page 2414)

Declared In

`NSSegmentedControl.h`

isEnabledForSegment:

Returns a Boolean value indicating whether the specified segment is enabled.

```
- (BOOL)isEnabledForSegment:(NSInteger)segment
```

Parameters

segment

The index of the segment whose enabled state you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Return Value

`YES` if the segment is enabled; otherwise, `NO`.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- setEnabled:forSegment:` (page 2413)

Declared In

`NSSegmentedControl.h`

isSelectedForSegment:

Returns a Boolean value indicating whether the specified segment is selected.

```
- (BOOL)isSelectedForSegment:(NSInteger)segment
```

Parameters

segment

The index of the segment whose selection state you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Return Value
`YES` if the segment is selected; otherwise, `NO`.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setSelected:forSegment:` (page 2417)

Declared In
`NSSegmentedControl.h`

labelForSegment:

Returns the label of the specified segment

```
- (NSString *)labelForSegment:(NSInteger)segment
```

Parameters
segment
The index of the segment whose label you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Return Value
The label of the segment. The returned string contains the entire text of the label, even if that text is normally truncated during drawing.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setLabel:forSegment:` (page 2415)

Declared In
`NSSegmentedControl.h`

menuForSegment:

Returns the menu for the specified segment

```
- (NSMenu *)menuForSegment:(NSInteger)segment
```

Parameters
segment
The index of the segment whose menu you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Return Value
The menu associated with the segment; otherwise, `nil`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setMenu:forSegment:` (page 2415)

Declared In
`NSSegmentedControl.h`

segmentCount

Returns the number of segments in the receiver.

```
- (NSInteger)segmentCount
```

Return Value
The number of segments.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setSegmentCount:` (page 2416)

Declared In
`NSSegmentedControl.h`

segmentStyle

Returns the visual style used to display the receiver.

```
- (NSSegmentStyle)segmentStyle
```

Return Value
An `NSSegmentStyle` value that specifies the visual display used by the receiver. For possible values see "Segmented Control Visual Styles" (page 2419).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setSegmentStyle:` (page 2416)

Declared In
`NSSegmentedControl.h`

selectedSegment

Returns the index of the selected segment of the receiver.

```
- (NSInteger)selectedSegment
```

Return Value
The index of the currently selected segment or -1 if no segment is selected. If the receiver allows multiple selections, this method returns the most recently selected segment.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setSelectedSegment:` (page 2418)

Related Sample Code
ClipboardViewer
ImageClient
ImageMap
ImageMapExample

Declared In
`NSSegmentedControl.h`

selectSegmentWithTag:

Selects the segment with the specified tag.

```
- (BOOL)selectSegmentWithTag:(NSInteger)tag
```

Parameters
tag
The tag associated with the desired segment.

Return Value
`YES` if the segment was selected successfully; otherwise, `NO`.

Discussion
Typically, you use Interface Builder to specify the tag for each segment. You may also set this value programmatically using the `setTag:forSegment:` method of `NSSegmentedCell`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTag:forSegment:` (page 2401) (NSSegmentedCell)

Declared In
`NSSegmentedControl.h`

setEnabled:forSegment:

Sets the enabled state of the specified segment

```
- (void)setEnabled:(BOOL)flag forSegment:(NSInteger)segment
```

Parameters
flag
`YES` to enable the segment; otherwise, `NO` to disable it.

segment
The index of the segment you want to enable or disable. This method raises an `NSRangeException` if the index is out of bounds.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- isEnabledForSegment:` (page 2410)

Related Sample Code
PDFKitLinker2

Declared In
`NSSegmentedControl.h`

setImage:forSegment:

Sets the image for the specified segment.

```
- (void)setImage:(NSImage *)image forSegment:(NSInteger)segment
```

Parameters
image
The image to apply to the segment or `nil` if you want to clear the existing image. Images are not scaled to fit inside a segment. If the image is larger than the available space, it is clipped.

segment
The index of the segment whose image you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- imageForSegment:` (page 2409)

Declared In
`NSSegmentedControl.h`

setImageScaling:forSegment:

Sets the scaling mode used to display the specified segment's image.

```
- (void)setImageScaling:(NSImageScaling)scaling forSegment:(NSInteger)segment
```

Parameters
scaling
One of the image scaling constants. For a list of possible values, see `NSImageScaling` (page 617).

segment
The index of the segment whose enabled state you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `imageForSegment:` (page 2409)

Declared In
`NSSegmentedControl.h`

setLabel:forSegment:

Sets the label for the specified segment.

```
- (void)setLabel:(NSString *)label forSegment:(NSInteger)segment
```

Parameters

label
The label you want to display in the segment. If the width of the string is greater than the width of the segment, the string's text is truncated during drawing.

segment
The index of the segment whose label you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `labelForSegment:` (page 2411)

Declared In
`NSSegmentedControl.h`

setMenu:forSegment:

Sets the menu for the specified segment.

```
- (void)setMenu:(NSMenu *)menu forSegment:(NSInteger)segment
```

Parameters

menu
The menu you want to add to the segment or `nil` to clear the current menu. This menu is displayed when the user clicks and holds the mouse button while the mouse is over the segment.

segment
The index of the segment whose menu you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Discussion
Adding a menu to a segment allows that segment to be used as a pop-up button.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `menuForSegment:` (page 2411)

Declared In
`NSSegmentedControl.h`

setSegmentCount:

Sets the number of segments in the receiver.

```
- (void)setSegmentCount:(NSInteger)count
```

Parameters

count

The number of segments the receiver should have. If this value is less than the number of segments currently in the receiver, segments are removed from the right of the control. Similarly, if the number is greater than the current number of segments, the new segments are added on the right. This value must be between `0` and `2049`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `segmentCount` (page 2412)

Declared In
`NSSegmentedControl.h`

setSegmentStyle:

Sets the visual style used to display the receiver.

```
- (void)setSegmentStyle:(NSSegmentStyle)segmentStyle
```

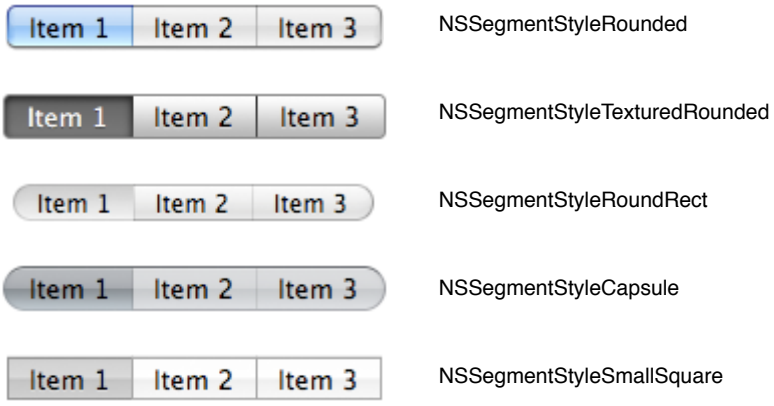
Parameters

segmentStyle

An `NSSegmentStyle` value that specifies the visual display used by the receiver. For the possible values see "Segmented Control Visual Styles" (page 2419).

Discussion
Figure 121-1 (page 2417) shows the visual styles and their corresponding `NSSegmentStyle` constant. The `NSSegmentStyleAutomatic` (page 2419) constant will automatically determined based on the type of window in which the control is displayed and the position within the window., and is not shown in the figure.

Figure 121-1 `NSSegmentStyle` examples



Availability
Available in Mac OS X v10.5 and later.

See Also
- `segmentStyle` (page 2412)

Declared In
`NSSegmentedControl.h`

setSelected:forSegment:

Sets the selection state of the specified segment.

```
- (void)setSelected:(BOOL)flag forSegment:(NSInteger)segment
```

Parameters

flag

`YES` if you want to select the segment; otherwise, `NO`.

segment

The index of the segment whose selection state you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Discussion
If the receiver allows only a single selection, this method deselects any other selected segments.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `isSelectedForSegment:` (page 2410)

Declared In
`NSSegmentedControl.h`

setSelectedSegment:

Sets the selected segment of the receiver.

```
- (void)setSelectedSegment:(NSInteger)selectedSegment
```

Parameters

selectedSegment

The zero-based index of the desired segment. This method raises an `NSRangeException` if the index is out of bounds.

Discussion

If the receiver allows multiple selections, this method selects the specified segment using `setSelected:forSegment:` (page 2417).

Availability

Available in Mac OS X v10.3 and later.

See Also

`- selectedSegment` (page 2412)

Related Sample Code

PDFKitLinker2

Declared In

`NSSegmentedControl.h`

setWidth:forSegment:

Sets the width of the specified segment.

```
- (void)setWidth:(CGFloat)width forSegment:(NSInteger)segment
```

Parameters

width

The width of the segment, measured in points. Specify the value `0` if you want the segment to be sized to fit the available space automatically.

segment

The index of the segment whose width you want to set. This method raises an `NSRangeException` if the index is out of bounds.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- widthForSegment:` (page 2418)

Declared In

`NSSegmentedControl.h`

widthForSegment:

Returns the width of the specified segment.

```
- (CGFloat)widthForSegment:(NSInteger)segment
```

Parameters

segment

The index of the segment whose width you want to get. This method raises an `NSRangeException` if the index is out of bounds.

Return Value

The width of the segment, measured in points, or 0 if the segment is sized to fit the available space automatically.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- setWidth:forSegment:` (page 2418)

Declared In

`NSSegmentedControl.h`

Constants

Segmented Control Visual Styles

The following constants specify the visual style used to display the segmented control. They are used by `setSegmentStyle:` (page 2416).

```
enum {
    NSSegmentStyleAutomatic = 0,
    NSSegmentStyleRounded = 1,
    NSSegmentStyleTexturedRounded = 2,
    NSSegmentStyleRoundRect = 3,
    NSSegmentStyleTexturedSquare = 4,
    NSSegmentStyleCapsule = 5,
    NSSegmentStyleSmallSquare = 6
};
typedef NSInteger NSSegmentStyle;
```

Constants

`NSSegmentStyleAutomatic`

The appearance of the segmented control is automatically determined based on the type of window in which the control is displayed and the position within the window.

Available in Mac OS X v10.5 and later.

Declared in `NSSegmentedControl.h`.

`NSSegmentStyleRounded`

The control is displayed using the rounded style. See Figure 121-1 (page 2417) for examples.

Available in Mac OS X v10.5 and later.

Declared in `NSSegmentedControl.h`.

`NSSegmentStyleTexturedRounded`
The control is displayed using the textured rounded style. See Figure 121-1 (page 2417) for examples.
Available in Mac OS X v10.5 and later.
Declared in `NSSegmentedControl.h`.

`NSSegmentStyleRoundRect`
The control is displayed using the round rect style. See Figure 121-1 (page 2417) for examples.
Available in Mac OS X v10.5 and later.
Declared in `NSSegmentedControl.h`.

`NSSegmentStyleTexturedSquare`
The control is displayed using the textured square style. See Figure 121-1 (page 2417) for examples.
Available in Mac OS X v10.5 and later.
Declared in `NSSegmentedControl.h`.

`NSSegmentStyleCapsule`
The control is displayed using the capsule style. See Figure 121-1 (page 2417) for examples.
Available in Mac OS X v10.5 and later.
Declared in `NSSegmentedControl.h`.

`NSSegmentStyleSmallSquare`
The control is displayed using the small square style. See Figure 121-1 (page 2417) for examples.
Available in Mac OS X v10.5 and later.
Declared in `NSSegmentedControl.h`.

NSShadow Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSShadow.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Cocoa Drawing Guide
Related sample code	FunHouse OpenCL NBody Simulation Example Reducer WebKitPluginStarter WebKitPluginWithJavaScript

Overview

An `NSShadow` object encapsulates the attributes used to create a drop shadow during drawing operations.

Shadows are always drawn in the default user coordinate space, regardless of any transformations applied to that space. This means that rotations, translations and other transformations of the current transformation matrix (the CTM) do not affect the resulting shadow. Another way to think about this is that changes to the CTM do not move or change the apparent position of the shadow's light source.

There are two positional parameters for a shadow: an x-offset and a y-offset. These values are expressed using a single `NSSize` data type and using the units of the default user coordinate space. Positive values for these offsets extend up and to the right.

In addition to its positional parameters, a shadow also contains a blur radius, which specifies how much a drawn object's image mask is blurred before it is composited onto the destination. A value of 0 means there is no blur. Larger values give correspondingly larger amounts of blurring.

An `NSShadow` object may be used in one of two ways. First, it may be set, like a color or a font, in which case its attributes are applied to all content drawn thereafter—or at least until another shadow is applied or a previous graphics state is restored. Second, it may be used as the value for the `NSShadowAttributeName` text attribute, in which case it is applied to the glyphs corresponding to the characters bearing this attribute.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

NSCopying
- `copyWithZone:`

Tasks

Creating a Shadow

- `init` (page 2423)
  Returns an `NSShadow` object initialized with default values.

Managing a Shadow

- `setShadowOffset:` (page 2424)
  Sets the offset values for the receiver.
- `shadowOffset` (page 2426)
  Returns the offset values for the receiver.
- `setShadowBlurRadius:` (page 2423)
  Sets the blur radius of the receiver.
- `shadowBlurRadius` (page 2425)
  Returns the blur radius of the receiver.
- `setShadowColor:` (page 2424)
  Sets the shadow color for the receiver.
- `shadowColor` (page 2425)
  Returns the color for the receiver.

Setting the Shadow

- `set` (page 2423)
  Sets the shadow of subsequent drawing operations to the shadow represented by the receiver.

Instance Methods

init

Returns an `NSShadow` object initialized with default values.

```
- (id)init
```

Return Value
An `NSShadow` object initialized with `0` as its offset, `0` as its blur radius, and the default color as its color. The returned object may be different from the original receiver.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSShadow.h`

set

Sets the shadow of subsequent drawing operations to the shadow represented by the receiver.

```
- (void)set
```

Discussion
The shadow attributes of the receiver are used until another shadow is set or until the graphics state is restored.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
DockTile
Reducer
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSShadow.h`

setShadowBlurRadius:

Sets the blur radius of the receiver.

```
- (void)setShadowBlurRadius:(CGFloat)val
```

Parameters

val

The blur radius, as measured in the default user coordinate space. A value of 0 indicates no blur, while larger values produce correspondingly larger blurring. This value must not be negative.

Availability
Available in Mac OS X v10.3 and later.

See Also
- shadowBlurRadius (page 2425)

Related Sample Code
ComplexBrowser
OpenCL NBody Simulation Example
Reducer
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSShadow.h

setShadowColor:

Sets the shadow color for the receiver.

```
- (void)setShadowColor:(NSColor *)color
```

Parameters

color

The shadow color, which must be convertible to an RGBA color. Specify nil if you do not want the shadow to be drawn. Your color may contain alpha information.

Availability
Available in Mac OS X v10.3 and later.

See Also
- shadowColor (page 2425)

Related Sample Code
DockTile
OpenCL NBody Simulation Example
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSShadow.h

setShadowOffset:

Sets the offset values for the receiver.

```
- (void)setShadowOffset:(NSSize)offset
```

Parameters

offset

The horizontal and vertical offset values, specified using the `width` and `height` fields of the `NSSize` data type. These offsets are measured using the default user coordinate space and are not affected by custom transformations. This means that positive values always extend up and to the right from the user's perspective.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- shadowOffset` (page 2426)

Related Sample Code
ComplexBrowser
OpenCL NBody Simulation Example
Reducer
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSShadow.h`

shadowBlurRadius

Returns the blur radius of the receiver.

```
- (CGFloat)shadowBlurRadius
```

Return Value
The blur radius, as measured in the default user coordinate space. A value of `0` indicates no blur, while larger values produce correspondingly larger blurring. The default value is `0`.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setShadowBlurRadius:` (page 2423)

Related Sample Code
FunHouse

Declared In
`NSShadow.h`

shadowColor

Returns the color for the receiver.

```
- (NSColor *)shadowColor
```

Return Value
The current shadow color. A `nil` shadow color indicates the shadow is not to be drawn. The default shadow color is black with an alpha of 1/3.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setShadowColor:` (page 2424)

Related Sample Code
FunHouse

Declared In
`NSShadow.h`

shadowOffset

Returns the offset values for the receiver.

```
- (NSSize)shadowOffset
```

Return Value
The horizontal and vertical offset values, specified using the `width` and `height` fields of the `NSSize` data type. These offsets are measured using the default user coordinate space and are not affected by custom transformations. This means that positive values always extend up and to the right from the user's perspective.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setShadowOffset:` (page 2424)

Related Sample Code
FunHouse

Declared In
`NSShadow.h`

NSSlider Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSSlider.h
Companion guide	Slider Programming Topics
Related sample code	Fireworks FunHouse MatrixMixerTest NineSlice OpenALExample

Overview

An `NSSlider` object displays a range of values for something in the application. Sliders can be vertical or horizontal bars or circular dials. An indicator, or knob, notes the current setting. The user can move the knob in the slider's bar—or rotate the knob in a circular slider—to change the setting.

The `NSSlider` class uses the `NSSliderCell` class to implement its user interface.

Tasks

Asking About the Slider's Appearance

- `altIncrementValue` (page 2430)
  Returns the amount by which the receiver changes its value when the user Option–drags the slider knob.
- `image` (page 2431)
  This method has been deprecated. Returns `nil`.

- `knobThickness` (page 2432)
  Returns the knob's thickness, in pixels.
- `isVertical` (page 2432)
  Returns an integer indicating the orientation (horizontal or vertical) of the slider.

Changing the Slider's Appearance

- `setAltIncrementValue:` (page 2435)
  Sets the amount by which the `NSSliderCell` modifies its value when the user Option-drags the knob.
- `setKnobThickness:` (page 2436)
  This method has been deprecated. Lets you set the knob's thickness, measured in pixels.
- `setImage:` (page 2435) Available in Mac OS X v10.0 through Mac OS X v10.5
  This method has been deprecated. Sets the image the receiver displays in the bar behind its knob.

Asking About the Slider's Title

- `title` (page 2441)
  Returns the receiver's title.
- `titleCell` (page 2441)
  This method has been deprecated. Returns `nil`.
- `titleColor` (page 2441)
  This method has been deprecated. Returns `nil`.
- `titleFont` (page 2442)
  This method has been deprecated. Returns `nil`.

Changing the Slider's Title

- `setTitle:` (page 2438)
  This method has been deprecated. Sets the title the receiver displays in the bar behind its knob.
- `setTitleCell:` (page 2439)
  This method has been deprecated. Sets the cell used to draw the receiver's title.
- `setTitleColor:` (page 2439)
  This method has been deprecated. Sets the color used to draw the receiver's title.
- `setTitleFont:` (page 2439)
  This method has been deprecated. Sets the font used to draw the receiver's title.

Asking About the Value Limits

- `maxValue` (page 2433)
  Returns the maximum value the receiver can send to its target.
- `minValue` (page 2433)
  Returns the minimum value the receiver can send to its target.

Changing the Value Limits

- `setMaxValue:` (page 2436)
  Sets the maximum value the receiver can send to its target.
- `setMinValue:` (page 2437)
  Sets the minimum value the receiver can send to its target

Handling Mouse-down Events

- `acceptsFirstMouse:` (page 2430)
  Returns a Boolean value indicating whether the slider accepts a single mouse-down event that simultaneously activates the window and takes hold of the slider's knob.

Managing Tick Marks

- `allowsTickMarkValuesOnly` (page 2430)
  Returns a Boolean value indicating whether the receiver fixes its values to those values represented by its tick marks.
- `closestTickMarkValueToValue:` (page 2431)
  Returns the value of the tick mark closest to the specified value.
- `indexOfTickMarkAtPoint:` (page 2432)
  Returns the index of the tick mark closest to the location of the receiver represented by the given point.
- `numberOfTickMarks` (page 2434)
  Returns the number of tick marks associated with the receiver.
- `rectOfTickMarkAtIndex:` (page 2434)
  Returns the bounding rectangle of the tick mark at the given index.
- `setAllowsTickMarkValuesOnly:` (page 2434)
  Sets whether the receiver's values are fixed to the values represented by the tick marks.
- `setNumberOfTickMarks:` (page 2437)
  Sets the number of tick marks displayed by the receiver.
- `setTickMarkPosition:` (page 2438)
  Sets where tick marks appear relative to the receiver.
- `tickMarkPosition` (page 2440)
  Returns how the receiver's tick marks are aligned with it.
- `tickMarkValueAtIndex:` (page 2440)
  Returns the receiver's value represented by the tick mark at the specified index.

Instance Methods

acceptsFirstMouse:

Returns a Boolean value indicating whether the slider accepts a single mouse-down event that simultaneously activates the window and takes hold of the slider's knob.

```
- (BOOL)acceptsFirstMouse:(NSEvent *)mouseDownEvent
```

Parameters

mouseDownEvent

The mouse-down event.

Return Value

`YES` if the receiver accepts the first mouse-down event; otherwise, `NO`. Returns `YES` by default.

Discussion

If you want the receiver to wait for its own mouse-down event, you must override this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSSlider.h`

allowsTickMarkValuesOnly

Returns a Boolean value indicating whether the receiver fixes its values to those values represented by its tick marks.

```
- (BOOL)allowsTickMarkValuesOnly
```

Return Value

`YES` if the slider fixes its values to the values represented by its tick marks; otherwise, `NO`.

Discussion

In its implementation of this method, the receiving `NSSlider` object simply invokes the method of the same name of its `NSSliderCell` object.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setAllowsTickMarkValuesOnly:` (page 2434)

Declared In

`NSSlider.h`

altIncrementValue

Returns the amount by which the receiver changes its value when the user Option–drags the slider knob.

```
- (double)altIncrementValue
```

Return Value
The amount by which the value changes when the user drags the slider knob with the Option key held down. Unless you call `setAltIncrementValue:` (page 2435), `altIncrementValue` (page 2430) returns –1.0, and the receiver behaves no differently with the Option key down than with it up.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAltIncrementValue:` (page 2435)

Declared In
`NSSlider.h`

closestTickMarkValueToValue:

Returns the value of the tick mark closest to the specified value.

```
- (double)closestTickMarkValueToValue:(double)aValue
```

Parameters
aValue
The value for which to return the closest tick mark.

Return Value
The value of the tick mark closest to `aValue`.

Discussion
In its implementation of this method, the receiver simply invokes the method of the same name of its `NSSliderCell` instance.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- indexOfTickMarkAtPoint:` (page 2432)

Declared In
`NSSlider.h`

image

This method has been deprecated. Returns `nil`.

```
- (NSImage *)image
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setImage:` (page 2435)

Declared In
NSSlider.h

indexOfTickMarkAtPoint:

Returns the index of the tick mark closest to the location of the receiver represented by the given point.

```
- (NSInteger)indexOfTickMarkAtPoint:(NSPoint)point
```

Parameters

point

The point representing the location for which to retrieve the tick mark.

Return Value
The index of the tick mark closest to the location specified by `point`. If *point* is not within the bounding rectangle (plus an extra pixel of space) of any tick mark, the method returns `NSNotFound`.

Discussion
In its implementation of this method, the receiving `NSSlider` instance simply invokes the method of the same name of its `NSSliderCell` instance. This method invokes `rectOfTickMarkAtIndex:` (page 2434) for each tick mark on the slider until it finds a tick mark containing the point.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- closestTickMarkValueToValue:` (page 2431)

Declared In
NSSlider.h

isVertical

Returns an integer indicating the orientation (horizontal or vertical) of the slider.

```
- (NSInteger)isVertical
```

Return Value
1 if the receiver is vertical, 0 if it's horizontal, and –1 if the orientation can't be determined (for example, if the slider hasn't been displayed yet). A slider is defined as vertical if its height is greater than its width.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSSlider.h

knobThickness

Returns the knob's thickness, in pixels.

```
- (CGFloat)knobThickness
```

Return Value
The thickness of the slider knob. The thickness is defined to be the extent of the knob along the long dimension of the bar. In a vertical slider, then, a knob's thickness is its height; in a horizontal slider, a knob's thickness is its width.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setKnobThickness:` (page 2436)

Declared In
`NSSlider.h`

maxValue

Returns the maximum value the receiver can send to its target.

```
- (double)maxValue
```

Return Value
The slider's maximum value. A horizontal slider sends its maximum value when the knob is at the right end of the bar; a vertical slider sends it when the knob is at the top.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMaxValue:` (page 2436)

Declared In
`NSSlider.h`

minValue

Returns the minimum value the receiver can send to its target.

```
- (double)minValue
```

Return Value
The slider's minimum value. A vertical slider sends its minimum value when its knob is at the bottom; a horizontal slider, when its knob is all the way to the left.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMinValue:` (page 2437)

Declared In
`NSSlider.h`

numberOfTickMarks

Returns the number of tick marks associated with the receiver.

```
- (NSInteger)numberOfTickMarks
```

Return Value
The number of the slider's tick marks.The tick marks assigned to the minimum and maximum values are included. In its implementation of this method, the receiving NNSSlider instance simply invokes the method of the same name of its NSSliderCell instance.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setNumberOfTickMarks: (page 2437)

Declared In
NSSlider.h

rectOfTickMarkAtIndex:

Returns the bounding rectangle of the tick mark at the given index.

```
- (NSRect)rectOfTickMarkAtIndex:(NSInteger)index
```

Parameters

index

The index of the tick mark for which to retrieve the bounds. The minimum-value tick mark is at index 0.

Return Value
The bounding rectangle of the specified tick mark.

Discussion
If no tick mark is associated with *index*, the method raises NSRangeException. In its implementation of this method, the receiving NSSlider instance simply invokes the method of the same name of its NSSliderCell instance.

Availability
Available in Mac OS X v10.0 and later.

See Also
- indexOfTickMarkAtPoint: (page 2432)

Declared In
NSSlider.h

setAllowsTickMarkValuesOnly:

Sets whether the receiver's values are fixed to the values represented by the tick marks.

```
- (void)setAllowsTickMarkValuesOnly:(BOOL)flag
```

Parameters

flag

YES if the slider's values should be fixed to the values represented by its tick marks; otherwise NO. For example, if a slider has a minimum value of 0, a maximum value of 100, and five markers, the allowable values are 0, 25, 50, 75, and 100. When users move the slider's knob, it jumps to the tick mark nearest the cursor when the mouse button is released.

Discussion

This method has no effect if the slider has no tick marks. In its implementation of this method, the receiving NSSlider instance simply invokes the method of the same name of its NSSliderCell instance.

Availability

Available in Mac OS X v10.0 and later.

See Also

- allowsTickMarkValuesOnly (page 2430)

Declared In

NSSlider.h

setAltIncrementValue:

Sets the amount by which the NSSliderCell modifies its value when the user Option-drags the knob.

```
- (void)setAltIncrementValue:(double)increment
```

Parameters

increment

The amount by which the slider's value changes when the user Option–drags its knob. This value must fit the range of values the slider can represent—for example, if the slider has a minimum value of 5 and a maximum value of 10, increment should be between 0 and 5. If *increment* is outside that range, the value is unchanged.

Discussion

If you don't call this method, the slider behaves the same with the Option key down as with it up. This is also the result when you call setAltIncrementValue: with an increment of –1.

Availability

Available in Mac OS X v10.0 and later.

See Also

- maxValue (page 2433)
- minValue (page 2433)

Declared In

NSSlider.h

setImage:

This method has been deprecated. Sets the image the receiver displays in the bar behind its knob.

```
- (void)setImage:(NSImage *)barImage
```

Parameters

barImage

The image to set.

Discussion

The slider may scale and distort *barImage* to fit inside the bar.

The knob may cover part of the image. If you want the image to be visible all the time, you're better off placing it near the slider.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setImage: (page 2435)

Declared In

NSSlider.h

setKnobThickness:

This method has been deprecated. Lets you set the knob's thickness, measured in pixels.

```
- (void)setKnobThickness:(CGFloat)thickness
```

Parameters

thickness

The thickness of the knob. The thickness is defined to be the extent of the knob along the long dimension of the bar. In a vertical slider, a knob's thickness is its height; in a horizontal slider, a knob's thickness is its width.

Availability

Available in Mac OS X v10.0 and later.

See Also

- knobThickness (page 2432)

Declared In

NSSlider.h

setMaxValue:

Sets the maximum value the receiver can send to its target.

```
- (void)setMaxValue:(double)maxValue
```

Parameters

maxValue

The maximum value of the slider. A horizontal slider sends its maximum value when its knob is all the way to the right; a vertical slider sends its maximum value when its knob is at the top.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `maxValue` (page 2433)

Related Sample Code
FunHouse
ImageApp

Declared In
`NSSlider.h`

setMinValue:

Sets the minimum value the receiver can send to its target

```
- (void)setMinValue:(double)minValue
```

Parameters

minValue

The minimum value of the slider. A horizontal slider sends its minimum value when its knob is all the way to the left; a vertical slider sends its minimum value when its knob is at the bottom.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `minValue` (page 2433)

Related Sample Code
FunHouse
ImageApp

Declared In
`NSSlider.h`

setNumberOfTickMarks:

Sets the number of tick marks displayed by the receiver.

```
- (void)setNumberOfTickMarks:(NSInteger)numberOfTickMarks
```

Parameters

numberOfTickMarks

The number of tick marks (including those assigned to the minimum and maximum values) displayed by the slider. By default, this value is 0, and no tick marks appear. The number of tick marks assigned to a slider, along with the slider's minimum and maximum values, determines the values associated with the tick marks.

Discussion
In its implementation of this method, the receiving `NSSlider` instance simply invokes the method of the same name of its `NSSliderCell` instance.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfTickMarks` (page 2434)

Declared In
`NSSlider.h`

setTickMarkPosition:

Sets where tick marks appear relative to the receiver.

```
- (void)setTickMarkPosition:(NSTickMarkPosition)position
```

Parameters

position

A constant indicating the position of the tick marks. For horizontal sliders, this can be `NSTickMarkBelow` (the default) or `NSTickMarkAbove`; for vertical sliders, this can be `NSTickMarkLeft` (the default) or `NSTickMarkRight`.

Discussion
This method has no effect if no tick marks have been assigned (that is, `numberOfTickMarks` (page 2434) returns 0). In its implementation of this method, the receiving `NSSlider` instance simply invokes the method of the same name of its `NSSliderCell` instance.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `tickMarkPosition` (page 2440)

Declared In
`NSSlider.h`

setTitle:

This method has been deprecated. Sets the title the receiver displays in the bar behind its knob.

```
- (void)setTitle:(NSString *)barTitle
```

Parameters

barTitle

The slider's title. The knob may cover part or all of the title. If you want the title to be visible all the time, you're better off placing a label near the slider.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `title` (page 2441)

Declared In
`NSSlider.h`

setTitleCell:

This method has been deprecated. Sets the cell used to draw the receiver's title.

```
- (void)setTitleCell:(NSCell *)titleCell
```

Parameters

titleCell

The cell used to draw the title.

Discussion

You only need to invoke this method if the default title cell, `NSTextFieldCell`, doesn't suit your needs—that is, you want to display the title in a manner that `NSTextFieldCell` doesn't permit. When you do choose to override the default, *titleCell* should be an instance of a subclass of `NSTextFieldCell`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- titleCell (page 2441)

Declared In

NSSlider.h

setTitleColor:

This method has been deprecated. Sets the color used to draw the receiver's title.

```
- (void)setTitleColor:(NSColor *)color
```

Parameters

color

The title color.

Availability

Available in Mac OS X v10.0 and later.

See Also

- titleColor (page 2441)

Declared In

NSSlider.h

setTitleFont:

This method has been deprecated. Sets the font used to draw the receiver's title.

```
- (void)setTitleFont:(NSFont *)font
```

Parameters

font

The title font.

Availability
Available in Mac OS X v10.0 and later.

See Also
- titleFont (page 2442)

Declared In
NSSlider.h

tickMarkPosition

Returns how the receiver's tick marks are aligned with it.

```
- (NSTickMarkPosition)tickMarkPosition
```

Return Value
A constant indicating the position of the tick marks. Possible values are NSTickMarkBelow, NSTickMarkAbove, NSTickMarkLeft, and NSTickMarkRight (the last two are for vertical sliders). The default alignments are NSTickMarkBelow and NSTickMarkLeft.

Discussion
In its implementation of this method, the receiving NSSlider instance simply invokes the method of the same name of its NSSliderCell instance.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTickMarkPosition: (page 2438)

Declared In
NSSlider.h

tickMarkValueAtIndex:

Returns the receiver's value represented by the tick mark at the specified index.

```
- (double)tickMarkValueAtIndex:(NSInteger)index
```

Parameters

index
The index of the tick mark for which to return the value. The minimum-value tick mark has an index of 0.

Return Value
The value of the specified tick mark.

Discussion
In its implementation of this method, the receiving NSSlider instance simply invokes the method of the same name of its NSSliderCell instance.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSSlider.h

title

Returns the receiver's title.

```
- (NSString *)title
```

Return Value
The title. The default title is the empty string (@"").

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTitle: (page 2438)

Declared In
NSSlider.h

titleCell

This method has been deprecated. Returns nil.

```
- (id)titleCell
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTitleCell: (page 2439)

Declared In
NSSlider.h

titleColor

This method has been deprecated. Returns nil.

```
- (NSColor *)titleColor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTitleColor: (page 2439)

Declared In
NSSlider.h

titleFont

This method has been deprecated. Returns `nil`.

```
- (NSFont *)titleFont
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitleFont:` (page 2439)

Declared In
`NSSlider.h`

NSSliderCell Class Reference

Inherits from	NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSSliderCell.h
Companion guide	Slider Programming Topics
Related sample code	AnimatedSlider QTKitMovieShuffler Sketch+Accessibility

Overview

An `NSSliderCell` object controls the appearance and behavior of an `NSSlider` object, or of a single slider in an `NSMatrix` object.

You can customize an `NSSliderCell` to a certain degree, using its `set...` methods. If these methods do not allow you sufficient flexibility, you can create a subclass. In that subclass, you can override any of the following methods: `knobRectFlipped:` (page 2450), `drawBarInside:flipped:` (page 2448), `drawKnob` (page 2448), and `prefersTrackingUntilMouseUp` (page 2446).

Tasks

Asking About the Cell's Behavior

`- altIncrementValue` (page 2447)

Returns the amount by which the slider changes its value when the user drags with the Option key held down.

`+ prefersTrackingUntilMouseUp` (page 2446)

Returns a Boolean value indicating whether the `NSSliderCell` continues to track the cursor until the next mouse up.

- `trackRect` (page 2459)
  Returns the rectangle within which the cell tracks the cursor while the mouse button is down.

Setting the Slider Type

- `setSliderType:` (page 2455)
  Sets the type of slider to a bar or a dial.
- `sliderType` (page 2457)
  Returns the slider type; either a bar or a dial.

Changing the Cell's Behavior

- `setAltIncrementValue:` (page 2452)
  Sets the amount by which the receiver modifies its value when the knob is Option-dragged.

Displaying the Cell

- `knobRectFlipped:` (page 2450)
  Returns the rectangle in which the slider knob is drawn.
- `drawBarInside:flipped:` (page 2448)
  Draws the slider's bar—but not its bezel or knob—inside the specified rectangle.
- `drawKnob` (page 2448)
  Calculates the rectangle in which the knob should be drawn, then invokes `drawKnob:` (page 2448) to actually draw the knob.
- `drawKnob:` (page 2448)
  Draws the slider knob in the given rectangle.

Asking About the Cell's Appearance

- `knobThickness` (page 2450)
  Returns the slider knob's thickness, in pixels.
- `isVertical` (page 2449)
  Returns an integer indicating the orientation (vertical or horizontal) of the slider.
- `title` (page 2458)
  This method has been deprecated. Returns the slider's title.
- `titleCell` (page 2459)
  This method has been deprecated. Returns `nil`.
- `titleFont` (page 2459)
  This method has been deprecated. Returns `nil`.
- `titleColor` (page 2459)
  This method has been deprecated. Returns `nil`.

Changing the Cell's Appearance

- `setKnobThickness:` (page 2453)
  This method has been deprecated. Lets you set the knob's thickness, measured in pixels.
- `setTitle:` (page 2456)
  This method has been deprecated. Sets the title in the bar behind the slider's knob.
- `setTitleCell:` (page 2456)
  This method has been deprecated. Sets the cell used to draw the slider's title.
- `setTitleColor:` (page 2456)
  This method has been deprecated. Sets the color used to draw the slider's title.
- `setTitleFont:` (page 2457)
  This method has been deprecated. Sets the font used to draw the slider's title.

Asking About the Value Limits

- `maxValue` (page 2450)
  Returns the maximum value the slider can send to its target.
- `minValue` (page 2451)
  Returns the minimum value the slider can send to its target.

Changing the Value Limits

- `setMaxValue:` (page 2453)
  Sets the maximum value the slider can send to its target.
- `setMinValue:` (page 2454)
  Sets the minimum value the slider can send to its target.

Managing Tick Marks

- `allowsTickMarkValuesOnly` (page 2446)
  Returns a Boolean value indicating whether the receiver fixes its values to those values represented by its tick marks.
- `closestTickMarkValueToValue:` (page 2447)
  Returns the value of the tick mark closest to the specified value.
- `indexOfTickMarkAtPoint:` (page 2449)
  Returns the index of the tick mark closest to the location of the slider represented by the specified point.
- `numberOfTickMarks` (page 2451)
  Returns the number of tick marks associated with the slider.
- `rectOfTickMarkAtIndex:` (page 2451)
  Returns the bounding rectangle of the tick mark at the specified index.
- `setAllowsTickMarkValuesOnly:` (page 2452)
  Sets whether the receiver's values are fixed to the values represented by the tick marks.

- `setNumberOfTickMarks:` (page 2454)
  Sets the number of tick marks displayed by the receiver.
- `setTickMarkPosition:` (page 2455)
  Sets where tick marks appear relative to the receiver.
- `tickMarkPosition` (page 2457)
  Returns the position of the tick marks relative to the receiver.
- `tickMarkValueAtIndex:` (page 2458)
  Returns the receiver's value represented by the tick mark at the specified index.

Class Methods

prefersTrackingUntilMouseUp

Returns a Boolean value indicating whether the `NSSliderCell` continues to track the cursor until the next mouse up.

```
+ (BOOL)prefersTrackingUntilMouseUp
```

Return Value
`YES` if the `NSSliderCell` continues to track the cursor even after the cursor leaves the cell's tracking rectangle; otherwise, `NO`. By default, this method returns `YES`.

Discussion
If this method returns `YES`, this means that, once you take hold of a slider's knob (by putting the cursor inside the cell's frame rectangle and pressing the mouse button), you retain control of the knob until you release the mouse button, even if you drag the cursor clear to the other side of the screen.

Never call this method explicitly. Override it if you create a subclass of `NSSliderCell` that you want to track the mouse differently.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSliderCell.h`

Instance Methods

allowsTickMarkValuesOnly

Returns a Boolean value indicating whether the receiver fixes its values to those values represented by its tick marks.

```
- (BOOL)allowsTickMarkValuesOnly
```

Return Value
`YES` if the slider's values are limited to those values represented by tick marks; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAllowsTickMarkValuesOnly:` (page 2452)

Declared In
`NSSliderCell.h`

altIncrementValue

Returns the amount by which the slider changes its value when the user drags with the Option key held down.

```
- (double)altIncrementValue
```

Return Value
The amount by which the slider changes its value when the user drags the knob with the Option key held down. Unless you call `setAltIncrementValue:` (page 2452), `altIncrementValue` (page 2447) returns –1.0, and the slider behaves no differently with the Option key down than with it up.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAltIncrementValue:` (page 2452)

Declared In
`NSSliderCell.h`

closestTickMarkValueToValue:

Returns the value of the tick mark closest to the specified value.

```
- (double)closestTickMarkValueToValue:(double)aValue
```

Parameters
aValue
The value for which to obtain the closest tick mark.

Return Value
The value of the closest tick mark.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfTickMarkAtPoint:` (page 2449)

Declared In
`NSSliderCell.h`

drawBarInside:flipped:

Draws the slider's bar—but not its bezel or knob—inside the specified rectangle.

```
- (void)drawBarInside:(NSRect)aRect flipped:(BOOL)flipped
```

Parameters

aRect

The bounds of the slider's bar, not of its interior rectangle.

flipped

A Boolean value that indicates whether the cell's control view—that is, the `NSSlider` or `NSMatrix` associated with the `NSSliderCell`—has a flipped coordinate system.

Discussion
You should never invoke this method explicitly. It's included so you can override it in a subclass.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- drawKnob:` (page 2448)

Declared In
`NSSliderCell.h`

drawKnob

Calculates the rectangle in which the knob should be drawn, then invokes `drawKnob:` (page 2448) to actually draw the knob.

```
- (void)drawKnob
```

Discussion
Before this message is sent, a `lockFocus` method must be sent to the cell's control view.

You might invoke this method if you override one of the display methods belonging to `NSControl` or `NSCell`.

If you create a subclass of `NSSliderCell`, don't override this method. Override `drawKnob:` (page 2448) instead.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSliderCell.h`

drawKnob:

Draws the slider knob in the given rectangle.

```
- (void)drawKnob:(NSRect)knobRect
```

Parameters

knobRect

The rectangle in which to draw the slider knob.

Discussion

Before this message is sent, a `lockFocus` (page 3187) message must be sent to the cell's control view.

You should never invoke this method explicitly. It's included so you can override it in a subclass.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSSliderCell.h`

indexOfTickMarkAtPoint:

Returns the index of the tick mark closest to the location of the slider represented by the specified point.

```
- (NSInteger)indexOfTickMarkAtPoint:(NSPoint)point
```

Parameters

point

The point representing the slider location.

Return Value

The index of the tick mark closest to the specified location.

Discussion

If *point* is not within the bounding rectangle (plus an extra pixel of space) of any tick mark, the method returns `NSNotFound`. This method invokes `rectOfTickMarkAtIndex:` (page 2451) for each tick mark on the slider until it finds a tick mark containing *point*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSSliderCell.h`

isVertical

Returns an integer indicating the orientation (vertical or horizontal) of the slider.

```
- (NSInteger)isVertical
```

Return Value

1 if the slider is vertical, 0 if it's horizontal, and –1 if the orientation can't be determined (for example, if the slider hasn't been displayed yet). A slider is defined as vertical if its height is greater than its width.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSSliderCell.h`

knobRectFlipped:

Returns the rectangle in which the slider knob is drawn.

```
- (NSRect)knobRectFlipped:(BOOL)flipped
```

Parameters

flipped

YES if the coordinate system of the associated NSSlider or NSMatrix is flipped; otherwise NO. You can determine whether this is the case by sending the NSView message isFlipped (page 3181) message to the NSMatrix or NSSlider.

Return Value

The rectangle in which the knob is drawn, specified in the coordinate system of the NSSlider or NSMatrix with which the receiver is associated.

The knob rectangle depends on where in the slider the knob belongs—that is, it depends on the receiver's minimum and maximum values and on the value the position of the knob will represent.

Discussion

You should never invoke this method explicitly. It's included so you can override it in a subclass.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSSliderCell.h

knobThickness

Returns the slider knob's thickness, in pixels.

```
- (CGFloat)knobThickness
```

Return Value

The thickness of the slider knob. The thickness is defined to be the extent of the knob along the long dimension of the bar. In a vertical slider, then, a knob's thickness is its height; in a horizontal slider, its thickness is its width.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setKnobThickness: (page 2453)

Declared In

NSSliderCell.h

maxValue

Returns the maximum value the slider can send to its target.

```
- (double)maxValue
```

Return Value
The maximum value of the slider. A horizontal slider sends its maximum value when the knob is at the right end of the slider; a vertical slider sends it when the knob is at the top. The maximum selectable value for a circular slider is just below `maxValue`; for example, if `maxValue` is 360, you can set the dial up to 359.999.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMaxValue:` (page 2453)

Declared In
`NSSliderCell.h`

minValue

Returns the minimum value the slider can send to its target.

```
- (double)minValue
```

Return Value
The minimum value of the slider. A vertical slider sends this value when its knob is at the bottom; a horizontal slider sends it when its knob is all the way to the left; a circular slider sends it when its knob is at the top.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSliderCell.h`

numberOfTickMarks

Returns the number of tick marks associated with the slider.

```
- (NSInteger)numberOfTickMarks
```

Return Value
The number of tick marks. The tick marks assigned to the minimum and maximum values are included.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setNumberOfTickMarks:` (page 2454)

Declared In
`NSSliderCell.h`

rectOfTickMarkAtIndex:

Returns the bounding rectangle of the tick mark at the specified index.

```
- (NSRect)rectOfTickMarkAtIndex:(NSInteger)index
```

Parameters

index

The index of the tick mark for which to return the bounding rectangle. The minimum-value tick mark is at index 0.

Return Value

The bounding rectangle of the specified tick mark.

Discussion

If no tick mark is associated with *index*, the method raises `NSRangeException`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- indexOfTickMarkAtPoint:` (page 2449)

Declared In

`NSSliderCell.h`

setAllowsTickMarkValuesOnly:

Sets whether the receiver's values are fixed to the values represented by the tick marks.

```
- (void)setAllowsTickMarkValuesOnly:(BOOL)flag
```

Parameters

flag

`YES` if the slider's values are fixed to the values represented by the slider's tick marks; otherwise `NO`. For example, if you specify `YES` for a slider that has a minimum value of 0, a maximum value of 100, and five markers, the allowable values are 0, 25, 50, 75, and 100. When users move the slider's knob, it jumps to the tick mark nearest the cursor when the mouse button is released. This method has no effect if the slider has no tick marks.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- allowsTickMarkValuesOnly` (page 2446)

Declared In

`NSSliderCell.h`

setAltIncrementValue:

Sets the amount by which the receiver modifies its value when the knob is Option-dragged.

```
- (void)setAltIncrementValue:(double)increment
```

Parameters

increment

The amount by which the receiver changes its value when the knob is Option-dragged. This number should t the range of values the slider can represent—for example, if the slider has a minimum value of 5 and a maximum value of 10, *increment* should be between 0 and 5.

Discussion

If you don't call this method, the slider behaves the same with the Option key down as with it up. This is also the result when you call `setAltIncrementValue:` (page 2452) with an increment of –1.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `maxValue` (page 2450)
- `minValue` (page 2451)

Declared In

`NSSliderCell.h`

setKnobThickness:

This method has been deprecated. Lets you set the knob's thickness, measured in pixels.

```
- (void)setKnobThickness:(CGFloat)thickness
```

Parameters

thickness

The knob's thickness. The thickness is defined to be the extent of the knob along the long dimension of the bar. In a vertical slider, then, a knob's thickness is its height; in a horizontal slider, its thickness is its width.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `knobThickness` (page 2450)

Declared In

`NSSliderCell.h`

setMaxValue:

Sets the maximum value the slider can send to its target.

```
- (void)setMaxValue:(double)aDouble
```

Parameters

aDouble

The slider's maximum value. A horizontal slider sends its maximum value when its knob is all the way to the right; a vertical slider sends its maximum value when its knob is at the top. The maximum selectable value for a circular slider is just below `maxValue`; for example, if `maxValue` is 360, you can set the dial up to 359.999.

Availability
Available in Mac OS X v10.0 and later.

See Also
- maxValue (page 2450)

Declared In
NSSliderCell.h

setMinValue:

Sets the minimum value the slider can send to its target.

```
- (void)setMinValue:(double)aDouble
```

Parameters

aDouble

The slider's minimum value. A horizontal slider sends its minimum value when its knob is all the way to the left; a vertical slider sends its minimum value when its knob is at the bottom; a circular slider sends it when its knob is at the top.

Availability
Available in Mac OS X v10.0 and later.

See Also
- minValue (page 2451)

Declared In
NSSliderCell.h

setNumberOfTickMarks:

Sets the number of tick marks displayed by the receiver.

```
- (void)setNumberOfTickMarks:(NSInteger)numberOfTickMarks
```

Parameters

numberOfTickMarks

The number of tick marks displayed by the slider, including those assigned to the minimum and maximum values. By default, this value is 0, and no tick marks appear. The number of tick marks assigned to a slider, along with the slider's minimum and maximum values, determines the values associated with the tick marks.

Availability
Available in Mac OS X v10.0 and later.

See Also
- numberOfTickMarks (page 2451)

Related Sample Code
QTKitMovieShuffler

Declared In
NSSliderCell.h

setSliderType:

Sets the type of slider to a bar or a dial.

```
- (void)setSliderType:(NSSliderType)sliderType
```

Parameters

sliderType

A constant indicating the type of the slider. Possible values are described in NSTickMarkPosition (page 2460).

Discussion
If sliderType is NSCircularSlider, then you get a fixed-size round slider. The minimum value (minValue) is at the top, and the value increases as you go clockwise around the dial. The maximum selectable value is just below maxValue; for example, if maxValue is 360, you can set the dial up to 359.999.

You can use the setNumberOfTickMarks: (page 2454) method to display tick marks, and you can use the setAllowsTickMarkValuesOnly: (page 2452) method to specify that values are limited to those values represented by tick marks. You can set this control to regular or small sizes; the mini size is not supported.

Availability
Available in Mac OS X v10.3 and later.

See Also
- sliderType (page 2457)
- setNumberOfTickMarks: (page 2454)
- setAllowsTickMarkValuesOnly: (page 2452)

Related Sample Code
QTKitMovieShuffler

Declared In
NSSliderCell.h

setTickMarkPosition:

Sets where tick marks appear relative to the receiver.

```
- (void)setTickMarkPosition:(NSTickMarkPosition)position
```

Parameters

position

A constant indicating the position of the tick marks. Possible values are described in NSTickMarkPosition (page 2460).

Discussion
This method has no effect if no tick marks have been assigned (that is, numberOfTickMarks (page 2451) returns 0).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `tickMarkPosition` (page 2457)

Declared In
`NSSliderCell.h`

setTitle:

This method has been deprecated. Sets the title in the bar behind the slider's knob.

```
- (void)setTitle:(NSString *)title
```

Parameters

title
The title.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `title` (page 2458)

Declared In
`NSSliderCell.h`

setTitleCell:

This method has been deprecated. Sets the cell used to draw the slider's title.

```
- (void)setTitleCell:(NSCell *)aCell
```

Discussion
You only need to invoke this method if the default title cell, `NSTextFieldCell`, doesn't suit your needs—that is, if you want to display the title in a manner that `NSTextFieldCell` doesn't permit. When you do choose to override the default, *aCell* should be an instance of a subclass of `NSTextFieldCell`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `titleCell` (page 2459)

Declared In
`NSSliderCell.h`

setTitleColor:

This method has been deprecated. Sets the color used to draw the slider's title.

```
- (void)setTitleColor:(NSColor *)color
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- titleColor (page 2459)

Declared In
NSSliderCell.h

setTitleFont:

This method has been deprecated. Sets the font used to draw the slider's title.

```
- (void)setTitleFont:(NSFont *)font
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- titleFont (page 2459)

Declared In
NSSliderCell.h

sliderType

Returns the slider type; either a bar or a dial.

```
- (NSSliderType)sliderType
```

Return Value
A constant indicating the type of the slider. Possible return values are described in NSSliderType (page 2461).

Availability
Available in Mac OS X v10.3 and later.

See Also
- setSliderType: (page 2455)

Declared In
NSSliderCell.h

tickMarkPosition

Returns the position of the tick marks relative to the receiver.

```
- (NSTickMarkPosition)tickMarkPosition
```

Return Value
A constant indicating the position of the tick marks. Possible values are described in `NSTickMarkPosition` (page 2460). The default alignments are `NSTickMarkBelow` and `NSTickMarkLeft`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTickMarkPosition:` (page 2455)

Declared In
`NSSliderCell.h`

tickMarkValueAtIndex:

Returns the receiver's value represented by the tick mark at the specified index.

```
- (double)tickMarkValueAtIndex:(NSInteger)index
```

Parameters
index
The index of the tick mark for which to retrieve the value. The minimum-value tick mark has an index of 0.

Return Value
The value represented by the specified tick mark.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSliderCell.h`

title

This method has been deprecated. Returns the slider's title.

```
- (NSString *)title
```

Return Value
The title. The default title is the empty string (@"").

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitle:` (page 2456)

Declared In
`NSSliderCell.h`

titleCell

This method has been deprecated. Returns `nil`.

```
- (id)titleCell
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitleCell:` (page 2456)

Declared In
`NSSliderCell.h`

titleColor

This method has been deprecated. Returns `nil`.

```
- (NSColor *)titleColor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitleColor:` (page 2456)

Declared In
`NSSliderCell.h`

titleFont

This method has been deprecated. Returns `nil`.

```
- (NSFont *)titleFont
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTitleFont:` (page 2457)

Declared In
`NSSliderCell.h`

trackRect

Returns the rectangle within which the cell tracks the cursor while the mouse button is down.

```
- (NSRect)trackRect
```

Return Value
The tracking rectangle of the `NSSliderCell`. This rectangle includes the slider bar, but not the bezel.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSliderCell.h`

Constants

NSTickMarkPosition

Specify where the tick marks of an `NSSliderCell` object appear.

```
typedef enum _NSTickMarkPosition {
    NSTickMarkBelow = 0,
    NSTickMarkAbove = 1,
    NSTickMarkLeft  = NSTickMarkAbove,
    NSTickMarkRight = NSTickMarkBelow
} NSTickMarkPosition;
```

Constants

`NSTickMarkBelow`
Tick marks below (for horizontal sliders); the default for horizontal sliders.
Available in Mac OS X v10.0 and later.
Declared in `NSSliderCell.h`.

`NSTickMarkAbove`
Tick marks above (for horizontal sliders).
Available in Mac OS X v10.0 and later.
Declared in `NSSliderCell.h`.

`NSTickMarkLeft`
Tick marks to the left (for vertical sliders); the default. for vertical sliders
Available in Mac OS X v10.0 and later.
Declared in `NSSliderCell.h`.

`NSTickMarkRight`
Tick marks to the right (for vertical sliders).
Available in Mac OS X v10.0 and later.
Declared in `NSSliderCell.h`.

Discussion
These constants are used in `setTickMarkPosition:` (page 2455) and `tickMarkPosition` (page 2457).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSliderCell.h`

NSSliderType

Define the types of sliders, used by `setSliderType:` (page 2455) and `sliderType` (page 2457).

```
typedef enum {
    NSLinearSlider   = 0,
    NSCircularSlider = 1
} NSSliderType;
```

Constants

`NSLinearSlider`

A bar-shaped slider.

Available in Mac OS X v10.3 and later.

Declared in `NSSliderCell.h`.

`NSCircularSlider`

A circular slider; that is, a dial.

Available in Mac OS X v10.3 and later.

Declared in `NSSliderCell.h`.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`NSSliderCell.h`

NSSound Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSPasteboardWriting NSPasteboardReading NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSSound.h
Companion guide	Sound Programming Topics for Cocoa
Related sample code	AttachAScript BundleLoader CustomSave GeekGameBoard TrackIt

Overview

The `NSSound` class provides a simple interface for loading and playing audio files. This class supports the same audio encodings and file formats that are supported by Core Audio and QuickTime.

To use this class, initialize a new instance with the desired file or audio data. You can configure assorted aspects of the audio playback, including the volume and whether the sound loops before you play it. Depending on the type of the audio data, this class may use either Core Audio or QuickTime to handle the actual playback. (Typically, it uses Core Audio to play files in the AIFF, WAVE, NeXT, SD2, AU, and MP3 formats and may use it for other formats in the future as well.) Playback occurs asynchronously so that your application can continue doing work.

You should retain `NSSound` objects before initiating playback or make sure you have a strong reference to them in a garbage-collected environment. Upon deallocation, a sound object stops playback of the sound (as needed) so that it can free up the corresponding audio resources. If you want to deallocate a sound object immediately after playback, assign a delegate and use the `sound:didFinishPlaying:` (page 3799) method to deallocate it.

If you want to play the system beep sound, use the `NSBeep` (page 3962) function.

Tasks

Creating Sounds

+ `canInitWithPasteboard:` (page 2466)
  Indicates whether the receiver can create an instance of itself from the data in a pasteboard.

- `initWithContentsOfFile:byReference:` (page 2470)
  Initializes the receiver with the the audio data located at a given filepath.

- `initWithContentsOfURL:byReference:` (page 2470)
  Initializes the receiver with the audio data located at a given URL.

- `initWithData:` (page 2470)
  Initializes the receiver with a given audio data.

- `initWithPasteboard:` (page 2471)
  Initializes the receiver with data from a pasteboard. The pasteboard should contain a type returned by `soundUnfilteredPasteboardTypes` (page 2467). `NSSound` expects the data to have a proper magic number, sound header, and data for the formats it supports.

Configuring Sounds

- `name` (page 2472)
  Returns the name assigned to the receiver.

- `setName:` (page 2475)
  Registers the receiver under a given name.

- `volume` (page 2477)
  Provides the volume of the receiver.

- `setVolume:` (page 2476)
  Specifies the volume of the receiver.

- `currentTime` (page 2469)
  Provides the receiver's playback progress in seconds.

- `setCurrentTime:` (page 2474)
  Specifies the receivers playback progress in seconds.

- `loops` (page 2472)
  Indicates whether the receiver restarts playback when it reaches the end of its content. Default: `NO`.

- `setLoops:` (page 2475)
  Specifies whether the receiver restarts playback when it reaches the end of its content.

- `playbackDeviceIdentifier` (page 2473)
  Identifies the receiver's output device.

- `setPlaybackDeviceIdentifier:` (page 2476)
  Specifies the receiver's output device.

- `channelMapping` (page 2468)
  Provides the receiver's channel map.

- `setChannelMapping:` (page 2474)
  Specifies the receiver's channel map.

- `delegate` (page 2469)
  Returns the receiver's delegate.
- `setDelegate:` (page 2474)
  Set the receiver's delegate.

Getting Sound Information

+ `soundUnfilteredTypes` (page 2468)
  Provides the file types the `NSSound` class understands.
+ `soundNamed:` (page 2466)
  Returns the `NSSound` instance associated with a given name.
- `duration` (page 2469)
  Provides the duration of the receiver in seconds.

Playing Sounds

- `isPlaying` (page 2471)
  Indicates whether the receiver is playing its audio data.
- `pause` (page 2472)
  Pauses audio playback.
- `play` (page 2473)
  Initiates audio playback.
- `resume` (page 2473)
  Resumes audio playback.
- `stop` (page 2477)
  Concludes audio playback.

Writing Sounds

- `writeToPasteboard:` (page 2477)
  Writes the receiver's data to a pasteboard.

Deprecated

+ `soundUnfilteredFileTypes` (page 2467) Deprecated in Mac OS X v10.5
  Provides the list of file types the `NSSound` class understands. (Deprecated. Use `soundUnfilteredTypes` (page 2468).)
+ `soundUnfilteredPasteboardTypes` (page 2467) Deprecated in Mac OS X v10.5
  Provides a list of the pasteboard types that the `NSSound` class can accept. (Deprecated. Use `soundUnfilteredTypes` (page 2468).)

Class Methods

canInitWithPasteboard:

Indicates whether the receiver can create an instance of itself from the data in a pasteboard.

```
+ (BOOL)canInitWithPasteboard:(NSPasteboard *)pasteboard
```

Parameters

pasteboard

Pasteboard containing sound data.

Return Value

`YES` when the receiver can handle the data represented by *pasteboard*; `NO` otherwise.

Discussion

The `soundUnfilteredPasteboardTypes` (page 2467) method is used to find out whether the class can handle the data in *pasteboard*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSSound.h`

soundNamed:

Returns the `NSSound` instance associated with a given name.

```
+ (id)soundNamed:(NSString *)soundName
```

Parameters

soundName

Name that identifies sound data.

Return Value

`NSSound` instance initialized with the sound data identified by *soundName*.

Discussion

The returned object can be one of the following:

- One that's been assigned a name with `setName:` (page 2475)
- One of the named system sounds provided by the Application Kit framework

If there's no known `NSSound` object with *soundName*, this method tries to create one by searching for sound files in the application's main bundle (see `NSBundle` for a description of how the bundle's contents are searched). If no sound file can be located in the application main bundle, the following directories are searched in order:

```
~/Library/Sounds
/Library/Sounds
/Network/Library/Sounds
```

```
/System/Library/Sounds
```

If no data can be found for *soundName*, no object is created, and `nil` is returned.

The preferred way to locate a sound is to pass a name without the file extension. See the class description for a list of the supported sound file extensions.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BundleLoader
ButtonMadness
CustomSave
GeekGameBoard
iChatTheater

Declared In
`NSSound.h`

soundUnfilteredFileTypes

Provides the list of file types the `NSSound` class understands. (Deprecated in Mac OS X v10.5. Use `soundUnfilteredTypes` (page 2468).)

```
+ (NSArray *)soundUnfilteredFileTypes
```

Return Value
Array of strings representing the file types the `NSSound` class understands.

Discussion
The returned array may be passed directly to the `runModalForTypes:` (page 1821) method of the `NSOpenPanel` class.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

Declared In
`NSSound.h`

soundUnfilteredPasteboardTypes

Provides a list of the pasteboard types that the `NSSound` class can accept. (Deprecated in Mac OS X v10.5. Use `soundUnfilteredTypes` (page 2468).)

```
+ (NSArray *)soundUnfilteredPasteboardTypes
```

Return Value
Array of pasteboard types that the `NSSound` class can accept.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

Declared In
`NSSound.h`

soundUnfilteredTypes

Provides the file types the `NSSound` class understands.

```
+ (NSArray *)soundUnfilteredTypes
```

Return Value
Array of UTIs identifying the file types the `NSSound` class understands.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSSound.h`

Instance Methods

channelMapping

Provides the receiver's channel map.

```
- (NSArray *)channelMapping
```

Return Value
The receiver's audio-channel–to–device–channel mappings.

Discussion
A **channel map** correlates a sound's channels to the the output-device's channels. For example, a two-channel sound being played on a five-channel device should have a channel map to optimize the sound-playing experience. The default map, correlates the first sound channel to the first output channel, the second sound channel to the second output channel, and so on.

For details about channel maps, see *Core Audio Overview* > "Common Tasks in Mac OS X."

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setChannelMapping:` (page 2474)

Declared In
`NSSound.h`

currentTime

Provides the receiver's playback progress in seconds.

```
- (NSTimeInterval)currentTime
```

Return Value
Receiver's playback progress in seconds.

Discussion
Sounds start with `currentTime == 0` and end with `currentTime == ([<sound> duration] - 1)`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setCurrentTime:` (page 2474)
- `duration` (page 2469)

Declared In
`NSSound.h`

delegate

Returns the receiver's delegate.

```
- (id < NSSoundDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDelegate:` (page 2474)

Declared In
`NSSound.h`

duration

Provides the duration of the receiver in seconds.

```
- (NSTimeInterval)duration
```

Return Value
Duration of the receiver in seconds.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSSound.h`

initWithContentsOfFile:byReference:

Initializes the receiver with the the audio data located at a given filepath.

```
- (id)initWithContentsOfFile:(NSString *)filepath byReference:(BOOL)byRef
```

Parameters

filepath

Path to the sound file with which the receiver is to be initialized.

byRef

When `YES` only the name of the sound is stored with the `NSSound` instance when archived using `encodeWithCoder:`; otherwise the audio data is archived along with the instance.

Return Value
Initialized `NSSound` instance.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AttachAScript

Declared In
`NSSound.h`

initWithContentsOfURL:byReference:

Initializes the receiver with the audio data located at a given URL.

```
- (id)initWithContentsOfURL:(NSURL *)fileUrl byReference:(BOOL)byRef
```

Parameters

fileUrl

URL to the sound file with which the receiver is to be initialized.

byRef

When `YES` only the name of the sound is stored with the `NSSound` instance when archived using `encodeWithCoder:`; otherwise the audio data is archived along with the instance.

Return Value
Initialized `NSSound` instance.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSound.h`

initWithData:

Initializes the receiver with a given audio data.

```
- (id)initWithData:(NSData *)audioData
```

Parameters

audioData

Audio data with which the receiver is to be initialized. The data must have a proper magic number, sound header, and data for the formats the `NSSound` class supports.

Return Value
Initialized `NSSound` instance.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSound.h`

initWithPasteboard:

Initializes the receiver with data from a pasteboard. The pasteboard should contain a type returned by `soundUnfilteredPasteboardTypes` (page 2467). `NSSound` expects the data to have a proper magic number, sound header, and data for the formats it supports.

```
- (id)initWithPasteboard:(NSPasteboard *)pasteboard
```

Parameters

pasteboard

The pasteboard containing the audio data with which the receiver is to be initialized. The pasteboard must contain a type returned by `soundUnfilteredPasteboardTypes` (page 2467). The contained data must have a proper magic number, sound header, and data for the formats the `NSSound` class supports.

Return Value
Initialized `NSSound` instance.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSound.h`

isPlaying

Indicates whether the receiver is playing its audio data.

```
- (BOOL)isPlaying
```

Return Value
`YES` when the receiver is playing its audio data, `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSound.h`

loops

Indicates whether the receiver restarts playback when it reaches the end of its content. Default: `NO`.

```
- (BOOL)loops
```

Return Value
`YES` when the receiver restarts playback when it finishes, `NO` otherwise.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setLoops:` (page 2475)

Declared In
`NSSound.h`

name

Returns the name assigned to the receiver.

```
- (NSString *)name
```

Return Value
Name assigned to the receiver; `nil` when no name has been assigned.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setName:` (page 2475)

Declared In
`NSSound.h`

pause

Pauses audio playback.

```
- (BOOL)pause
```

Return Value
`YES` when playback is paused successfully, `NO` when playback is already paused or when an error occurred.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSound.h`

play

Initiates audio playback.

```
- (BOOL)play
```

Return Value
`YES` when playback is initiated, `NO` when playback is already in progress or when an error occurred.

Discussion
This method initiates playback asynchronously and returns control to your application. Therefore, your application can continue doing work while the audio is playing.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AttachAScript
BundleLoader
CocoaSlides

Declared In
`NSSound.h`

playbackDeviceIdentifier

Identifies the receiver's output device.

```
- (NSString *)playbackDeviceIdentifier
```

Return Value
Unique identifier of a sound output device.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setPlaybackDeviceIdentifier:` (page 2476)

Declared In
`NSSound.h`

resume

Resumes audio playback.

```
- (BOOL)resume
```

Return Value
`YES` when playback is resumed, `NO` when playback is in progress or when an error occurred.

Discussion
Assumes the receiver has been previously paused by sending it `pause` (page 2472).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSSound.h

setChannelMapping:

Specifies the receiver's channel map.

```
- (void)setChannelMapping:(NSArray *)channelMapping
```

Parameters

channelMapping
Audio-channel—to—device–channel mappings for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- channelMapping (page 2468)

Declared In
NSSound.h

setCurrentTime:

Specifies the receivers playback progress in seconds.

```
- (void)setCurrentTime:(NSTimeInterval)currentTime
```

Parameters

currentTime
Playback progress for the receiver.

Discussion
This property is not archived, copied, or stored on the pasteboard.

Availability
Available in Mac OS X v10.5 and later.

See Also
- currentTime (page 2469)

Declared In
NSSound.h

setDelegate:

Set the receiver's delegate.

```
- (void)setDelegate:(id < NSSoundDelegate >)delegate
```

Parameters

delegate

Object to serve as the receiver's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- delegate (page 2469)

Declared In
NSSound.h

setLoops:

Specifies whether the receiver restarts playback when it reaches the end of its content.

```
- (void)setLoops:(BOOL)loops
```

Parameters

Term

YES to have the receiver restart playback when it reaches the end of its content.

NO to have the receiver conclude playback, instead.

Discussion
When *loops* is YES, the receiver does not send sound:didFinishPlaying: (page 3799) to its delegate when it reaches the end of its content and restarts playback.

Availability
Available in Mac OS X v10.5 and later.

See Also
- loops (page 2472)
- stop (page 2477)

Declared In
NSSound.h

setName:

Registers the receiver under a given name.

```
- (BOOL)setName:(NSString *)soundName
```

Parameters

soundName

Name to assign the receiver. The name must be unused by other NSSound instances.

Return Value
YES when successful; NO otherwise.

Discussion
If the receiver is already registered under another name, this method first unregisters the prior name.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- name` (page 2472)
- `+ soundNamed:` (page 2466)

Declared In
`NSSound.h`

setPlaybackDeviceIdentifier:

Specifies the receiver's output device.

```
- (void)setPlaybackDeviceIdentifier:(NSString *)playbackDeviceIdentifier
```

Parameters

playbackDeviceIdentifier
Unique identifier of a sound output device.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `- playbackDeviceIdentifier` (page 2473)

Declared In
`NSSound.h`

setVolume:

Specifies the volume of the receiver.

```
- (void)setVolume:(float)volume
```

Parameters

volume
Volume at which the receiver is to play.

Discussion
The valid range is between 0.0 and 1.0.

This method does not affect the systemwide volume.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `- volume` (page 2477)

Declared In
`NSSound.h`

stop

Concludes audio playback.

```
- (BOOL)stop
```

Return Value
`YES` when playback is concluded successfully or if it's paused, `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- sound:didFinishPlaying:` (page 3799) (NSSoundDelegate)

Declared In
`NSSound.h`

volume

Provides the volume of the receiver.

```
- (float)volume
```

Return Value
Volume of the receiver.

Discussion
The valid range is between 0.0 and 1.0.

This method does not affect the systemwide volume.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setVolume:` (page 2476)

Declared In
`NSSound.h`

writeToPasteboard:

Writes the receiver's data to a pasteboard.

```
- (void)writeToPasteboard:(NSPasteboard *)pasteboard
```

Parameters
pasteboard
Pasteboard to which the receiver is to write its data.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSound.h`

Constants

NSPasteboard Type for Sound Data

The `NSSound` class defines this common pasteboard data type.

```
NSString *NSSoundPboardType;
```

Constants
`NSSoundPboardType`
`NSSound` data

Available in Mac OS X v10.0 and later.

Declared in `NSSound.h`.

NSSpeechRecognizer Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.3 and later.
Declared in	AppKit/NSSpeechRecognizer.h
Companion guide	Speech

Overview

The NSSpeechRecognizer class is the Cocoa interface to Speech Recognition on Mac OS X. Speech Recognition is architected as a "command and control" voice recognition system. It uses a finite state grammar and listens for phrases in that grammar. When it recognizes a phrase, it notifies the client process. This architecture is different from that used to support dictation.

Through an NSSpeechRecognizer instance, Cocoa applications can use the speech recognition engine built into Mac OS X to recognize spoken commands. With speech recognition, users can accomplish complex, multi-step tasks with one spoken command—for example, "schedule a meeting with Adam and John tomorrow at ten o'clock."

The NSSpeechRecognizer class has methods that let you specify which spoken words should be recognized as commands (`setCommands:` (page 2483)) and to start and stop listening (`startListening` (page 2484) and `stopListening` (page 2485)). When the Speech Recognition facility recognizes one of the designated commands, NSSpeechRecognizer invokes the delegation method `speechRecognizer:didRecognizeCommand:` (page 3801), allowing the delegate to perform the command.

Speech Recognition is just one of the Mac OS X speech technologies. The Speech Synthesis technology allows applications to "pronounce" written text in U.S. English; the NSSpeechSynthesizer class is the Cocoa interface to this technology. These technologies provide benefits for all users, and are particularly useful to those users who have difficulties seeing the screen or using the mouse and keyboard. By incorporating speech into your application, you can provide a concurrent mode of interaction for your users: In Mac OS X, your software can accept input and provide output without requiring users to change their working context.

Tasks

Creating Speech Recognizers

- `init` (page 2482)
  Initializes and returns an instance of the NSSpeechRecognizer class.

Configuring Speech Recognizers

- `commands` (page 2481)
  Returns an array of strings defining the commands for which the receiver should listen.
- `setCommands:` (page 2483)
  Sets the list of commands for which the receiver should listen to *`commands`*.
- `displayedCommandsTitle` (page 2482)
  Returns the title of the commands section or `nil` if there is no title.
- `setDisplayedCommandsTitle:` (page 2484)
  Sets whether the speech-recognition commands should be displayed indented under a section title in the Speech Commands window, and if so, sets the title string to display.
- `listensInForegroundOnly` (page 2482)
  Returns whether the receiver should only enable its commands when the receiver's application is the frontmost one.
- `setListensInForegroundOnly:` (page 2484)
  Sets whether the receiver should only enable its commands when the receiver's application is the frontmost one.
- `blocksOtherRecognizers` (page 2481)
  Returns whether the receiver should block all other recognizers (that is, other applications attempting to understand spoken commands) when listening.
- `setBlocksOtherRecognizers:` (page 2482)
  Sets whether the receiver's commands should be the only enabled commands on the system.
- `delegate` (page 2481)
  Returns the receiver's delegate.
- `setDelegate:` (page 2483)
  Sets the receiver's delegate.

Listening

- `startListening` (page 2484)
  Tells the speech recognition engine to begin listening for commands.
- `stopListening` (page 2485)
  Tells the speech recognition engine to suspend listening for commands.

Instance Methods

blocksOtherRecognizers

Returns whether the receiver should block all other recognizers (that is, other applications attempting to understand spoken commands) when listening.

```
- (BOOL)blocksOtherRecognizers
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setBlocksOtherRecognizers:` (page 2482)

Declared In
`NSSpeechRecognizer.h`

commands

Returns an array of strings defining the commands for which the receiver should listen.

```
- (NSArray *)commands
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setCommands:` (page 2483)

Declared In
`NSSpeechRecognizer.h`

delegate

Returns the receiver's delegate.

```
- (id < NSSpeechRecognizerDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setDelegate:` (page 2483)

Declared In
`NSSpeechRecognizer.h`

displayedCommandsTitle

Returns the title of the commands section or `nil` if there is no title.

```
- (NSString *)displayedCommandsTitle
```

Discussion
Commands are displayed in the Speech Commands window indented under a section with this title.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setDisplayedCommandsTitle:` (page 2484)

Declared In
`NSSpeechRecognizer.h`

init

Initializes and returns an instance of the NSSpeechRecognizer class.

```
- (id)init
```

Discussion
Returns `nil` if initialization did not succeed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSSpeechRecognizer.h`

listensInForegroundOnly

Returns whether the receiver should only enable its commands when the receiver's application is the frontmost one.

```
- (BOOL)listensInForegroundOnly
```

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setListensInForegroundOnly:` (page 2484)

Declared In
`NSSpeechRecognizer.h`

setBlocksOtherRecognizers:

Sets whether the receiver's commands should be the only enabled commands on the system.

```
- (void)setBlocksOtherRecognizers:(BOOL)flag
```

Discussion
If *flag* is `YES`, all other speech recognition commands on the system are disabled until the receiver object is released, listening is stopped, or this method is called again with *flag* as `NO`. Because this option effectively takes over the computer at the expense of other applications using speech recognition, you should use it only in circumstances that warrant it, such as when listening for a response important to overall system operation or when an application is running in full-screen mode (such as games and presentation software). The default is `NO`.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- blocksOtherRecognizers` (page 2481)

Declared In
`NSSpeechRecognizer.h`

setCommands:

Sets the list of commands for which the receiver should listen to *commands*.

```
- (void)setCommands:(NSArray *)commands
```

Discussion
If the receiver is already listening, the current command list is updated and listening continues. *commands* should be an array of NSString objects. The commands must be in U.S. English.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- commands` (page 2481)

Declared In
`NSSpeechRecognizer.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSSpeechRecognizerDelegate >)anObject
```

Parameters
anObject
The delegate to set as the receiver's. The delegate must conform to the `NSSpeechRecognizerDelegate Protocol` protocol.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `delegate` (page 2481)

Declared In
`NSSpeechRecognizer.h`

setDisplayedCommandsTitle:

Sets whether the speech-recognition commands should be displayed indented under a section title in the Speech Commands window, and if so, sets the title string to display.

```
- (void)setDisplayedCommandsTitle:(NSString *)title
```

Discussion
When *title* is a non-empty string, the receiver's commands are displayed under a section with *title*. If *title* is `nil` or an empty string, the commands are displayed at the top level of the Speech Commands window. This default is not to display the commands under a section title.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `displayedCommandsTitle` (page 2482)

Declared In
`NSSpeechRecognizer.h`

setListensInForegroundOnly:

Sets whether the receiver should only enable its commands when the receiver's application is the frontmost one.

```
- (void)setListensInForegroundOnly:(BOOL)flag
```

Discussion
If *flag* is `YES`, the receiver's commands are only recognized when the receiver's application is the frontmost application—normally the application displaying the menu bar. If *flag* is `NO`, the commands are recognized regardless of the visibility of applications, including agent applications (agent applications, which have the `LSUIElement` property set, do not appear in the Dock or Force Quit window). The default is `YES`.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `listensInForegroundOnly` (page 2482)

Declared In
`NSSpeechRecognizer.h`

startListening

Tells the speech recognition engine to begin listening for commands.

```
- (void)startListening
```

Discussion
When a command is recognized the message `speechRecognizer:didRecognizeCommand:` (page 3801) is sent to the delegate.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- stopListening` (page 2485)

Declared In
`NSSpeechRecognizer.h`

stopListening

Tells the speech recognition engine to suspend listening for commands.

```
- (void)stopListening
```

Availability
Available in Mac OS X v10.3 and later.

See Also
`- startListening` (page 2484)

Declared In
`NSSpeechRecognizer.h`

NSSpeechSynthesizer Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSSpeechSynthesizer.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Speech
Related sample code	SayIt

Overview

The `NSSpeechSynthesizer` class is the Cocoa interface to Speech Synthesis on Mac OS X. Instances of this class are called **speech synthesizers**.

Speech Synthesis, also called text-to-speech (TTS), parses text and converts it into audible speech. It offers a concurrent feedback mode that can be used in concert with or in place of traditional visual and aural notifications. For example, your application can use an `NSSpeechSynthesizer` object to "pronounce" the text of important alert dialogs. Synthesized speech has several advantages. It can provide urgent information to users without forcing them to shift attention from their current task. And because speech doesn't rely on visual elements for meaning, it is a crucial technology for users with vision or attention disabilities.

In addition, synthesized speech can help save system resources. Because sound samples can take up large amounts of room on disk, using text in place of sampled sound is extremely efficient, and so a multimedia application might use an `NSSpeechSynthesizer` object to provide a narration of a QuickTime movie instead of including sampled-sound data on a movie track.

When you create an `NSSpeechSynthesizer` instance using the default initializer (`init`), the class uses the **default voice** selected in System Preferences > Speech. Alternatively, you can select a specific voice for an `NSSpeechSynthesizer` instance by initializing it with `initWithVoice:` (page 2493). To begin synthesis, send either `startSpeakingString:` (page 2498) or `startSpeakingString:toURL:` (page 2499) to the instance. The former generates speech through the system's default sound output device; the latter saves the generated speech to a file. If you wish to be notified when the current speech concludes, set a delegate object using `setDelegate:` (page 2496) and implement the delegate method `speechSynthesizer:didFinishSpeaking:` (page 3805).

Speech Synthesis is just one of the Mac OS X speech technologies. The Speech Recognizer technology allows applications to "listen to" text spoken in U.S. English; the `NSSpeechRecognizer` class is the Cocoa interface to this technology. Both technologies provide benefits for all users, and are particularly useful to those users who have difficulties seeing the screen or using the mouse and keyboard.

Speech Feedback Window

The speech feedback window (Figure 127-1) displays the text recognized from the user's speech and the text from which an `NSSpeechSynthesizer` object synthesizes speech. Using the feedback window makes spoken exchange more natural and helps the user understand the synthesized speech.

Figure 127-1 Speech feedback window

For example, your application may use an `NSSpeechRecognizer` object to listen for the command "Play some music." When it recognizes this command, your application might then respond by speaking "Which artist?" using a speech synthesizer.

When `UsesFeedbackWindow` is `YES`, the speech synthesizer uses the feedback window if its visible, which the user specifies in System Preferences > Speech.

Tasks

Creating Speech Synthesizers

`- initWithVoice:` (page 2493)
Initializes the receiver with a voice.

Configuring Speech Synthesizers

`- usesFeedbackWindow` (page 2501)
Indicates whether the receiver uses the speech feedback window.

`- setUsesFeedbackWindow:` (page 2497)
Specifies whether the receiver uses the speech feedback window.

- `voice` (page 2501)
  Returns the identifier of the receiver's current voice.
- `setVoice:` (page 2497)
  Sets the receiver's current voice.
- `rate` (page 2495)
  Provides the receiver's speaking rate.
- `setRate:` (page 2496)
  Specifies the receivers speaking rate.
- `volume` (page 2501)
  Provides the receiver's speaking volume.
- `setVolume:` (page 2498)
  Specifies the receiver's speaking volume.
- `addSpeechDictionary:` (page 2491)
  Registers the given speech dictionary with the receiver.
- `objectForProperty:error:` (page 2494)
  Provides the value of a receiver's property.
- `setObject:forProperty:error:` (page 2496)
  Specifies the value of a receiver's property.
- `delegate` (page 2492)
  Returns the receiver's delegate.
- `setDelegate:` (page 2496)
  Sets the receiver's delegate.

Getting Speech Synthesizer Information

+ `availableVoices` (page 2490)
  Provides the identifiers of the voices available on the system.
+ `attributesForVoice:` (page 2490)
  Provides the attribute dictionary of a voice.
+ `defaultVoice` (page 2491)
  Provides the identifier of the default voice.

Getting Speech State

+ `isAnyApplicationSpeaking` (page 2491)
  Indicates whether any application is currently speaking through the sound output device.

Synthesizing Speech

- `isSpeaking` (page 2493)
  Indicates whether the receiver is currently generating synthesized speech.
- `startSpeakingString:` (page 2498)
  Begins speaking synthesized text through the system's default sound output device.

- `startSpeakingString:toURL:` (page 2499)
  Begins synthesizing text into a sound (AIFF) file.
- `pauseSpeakingAtBoundary:` (page 2494)
  Pauses synthesis in progress at a given boundary.
- `continueSpeaking` (page 2492)
  Resumes synthesis.
- `stopSpeaking` (page 2500)
  Stops synthesis in progress.
- `stopSpeakingAtBoundary:` (page 2500)
  Stops synthesis in progress at a given boundary.

Getting Phonemes

- `phonemesFromText:` (page 2495)
  Provides the phoneme symbols generated by the given text.

Class Methods

attributesForVoice:

Provides the attribute dictionary of a voice.

```
+ (NSDictionary *)attributesForVoice:(NSString *)voiceIdentifier
```

Parameters

voiceIdentifier
  Identifier of the voice whose attributes you want to obtain.

Return Value
Attribute dictionary of the voice identified by *voiceIdentifier*. The attributes keys and value types are listed in "Voice Attributes" (page 2502)

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSSpeechSynthesizer.h

availableVoices

Provides the identifiers of the voices available on the system.

```
+ (NSArray *)availableVoices
```

Return Value
Array of strings representing the identifiers of each voice available on the system.

Availability
Available in Mac OS X v10.3 and later.

See Also
`+ attributesForVoice:` (page 2490)
`- setVoice:` (page 2497)

Declared In
`NSSpeechSynthesizer.h`

defaultVoice

Provides the identifier of the default voice.

```
+ (NSString *)defaultVoice
```

Return Value
Identifier of the default voice.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSSpeechSynthesizer.h`

isAnyApplicationSpeaking

Indicates whether any application is currently speaking through the sound output device.

```
+ (BOOL)isAnyApplicationSpeaking
```

Return Value
`YES` when another application is producing speech through the sound output device, `NO` otherwise.

Discussion
You usually invoke this method to prevent your application from speaking over speech being generated by another application or system component.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSSpeechSynthesizer.h`

Instance Methods

addSpeechDictionary:

Registers the given speech dictionary with the receiver.

```
- (void)addSpeechDictionary:(NSDictionary *)speechDictionary
```

Parameters

speechDictionary

Speech dictionary to add to the receiver's dictionaries. The key-value pairs are listed in "Speech Dictionary Properties" (page 2513).

Discussion
See the discussion of `UseSpeechDictionary` in *Speech Synthesis Manager Reference* for more information.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSSpeechSynthesizer.h`

continueSpeaking

Resumes synthesis.

```
- (void)continueSpeaking
```

Discussion
At any time after `pauseSpeakingAtBoundary:` (page 2494) is called, `continueSpeaking` (page 2492) can be called to continue speaking from the beginning of the word at which speech paused.

Sending `continueSpeaking` (page 2492) to a receiver that is not currently in a paused state has no effect on the synthesizer or on future calls to the `pauseSpeakingAtBoundary:` (page 2494) function. If you call `continueSpeaking` (page 2492) on a synthesizer before a pause is effective, `continueSpeaking` (page 2492) cancels the pause.

If the `pauseSpeakingAtBoundary:` (page 2494) method stopped speech in the middle of a word, the synthesizer will start speaking that word from the beginning when you call `continueSpeaking` (page 2492).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `pauseSpeakingAtBoundary:` (page 2494)

Declared In
`NSSpeechSynthesizer.h`

delegate

Returns the receiver's delegate.

```
- (id < NSSpeechSynthesizerDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setDelegate:` (page 2496)

Declared In
`NSSpeechSynthesizer.h`

initWithVoice:

Initializes the receiver with a voice.

```
- (id)initWithVoice:(NSString *)voiceIdentifier
```

Parameters

voiceIdentifier
: Identifier of the voice to set as the current voice. When `nil`, the default voice is used. Passing in a specific voice means the initial speaking rate is determined by the synthesizer's default speaking rate; passing `nil` means the speaking rate is automatically set to the rate the user specifies in Speech preferences.

Return Value
Initialized speech synthesizer or `nil` when the voice identified by *voiceIdentifier* is not available or when there's an allocation error.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `+ availableVoices` (page 2490)

Declared In
`NSSpeechSynthesizer.h`

isSpeaking

Indicates whether the receiver is currently generating synthesized speech.

```
- (BOOL)isSpeaking
```

Return Value
`YES` when the receiver is generating synthesized speech, `NO` otherwise.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `startSpeakingString:` (page 2498)
- `startSpeakingString:toURL:` (page 2499)
- `stopSpeaking` (page 2500)

Declared In
`NSSpeechSynthesizer.h`

objectForProperty:error:

Provides the value of a receiver's property.

```
- (id)objectForProperty:(NSString *)speechProperty error:(NSError **)out_error
```

Parameters

speechProperty

Property to get.

out_error

On output, error that occurred while obtaining the value of *speechProperty*.

Return Value

The value of *speechProperty*.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `setObject:forProperty:error:` (page 2496)

Declared In

`NSSpeechSynthesizer.h`

pauseSpeakingAtBoundary:

Pauses synthesis in progress at a given boundary.

```
- (void)pauseSpeakingAtBoundary:(NSSpeechBoundary)boundary
```

Parameters

boundary

Boundary at which to pause speech. The supported bound types are listed in "NSSpeechBoundary" (page 2514).

Discussion

Pass the constant `NSSpeechImmediateBoundary` (page 2514) to pause immediately, even in the middle of a word. Pass `NSSpeechWordBoundary` (page 2514) or `NSSpeechSentenceBoundary` (page 2514) to pause speech at the end of the current word or sentence, respectively.

You can determine whether your application has paused a synthesizer's speech output by obtaining the `NSSpeechStatusProperty` (page 2505) property through the `objectForProperty:error:` (page 2494) method. While a synthesizer is paused, the speech status information indicates that `NSSpeechStatusOutputBusy` (page 2510) and `NSSpeechStatusOutputPaused` (page 2510) are both `YES`.

If the end of the string being spoken is reached before the specified pause point, speech output pauses at the end of the string.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `continueSpeaking` (page 2492)

Declared In
NSSpeechSynthesizer.h

phonemesFromText:

Provides the phoneme symbols generated by the given text.

```
- (NSString *)phonemesFromText:(NSString *)text
```

Parameters

text

Text from which to generate phonemes.

Return Value
Phonemes generated from text.

Discussion
Converting textual data into phonetic data is particularly useful during application development, when you might wish to adjust phrases that your application generates to produce smoother speech. By first converting the target phrase into phonemes, you can see what the synthesizer will try to speak. Then you need correct only the parts that would not have been spoken the way you want

The string returned by phonemesFromText: corresponds precisely to the phonemes that would be spoken had the input text been sent to `startSpeakingString:` (page 2498) instead. All current property settings for the synthesizer are applied to the converted speech.

No delegate methods are called while `phonemesFromText:` (page 2495) method is generating its output.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSSpeechSynthesizer.h

rate

Provides the receiver's speaking rate.

```
- (float)rate
```

Return Value
Speaking rate (words per minute).

Discussion
The range of supported rates is not predefined by the Speech Synthesis framework; but the synthesizer may only respond to a limited range of speech rates. Average human speech occurs at a rate of 180 to 220 words per minute.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setRate:` (page 2496)

Declared In
NSSpeechSynthesizer.h

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSSpeechSynthesizerDelegate >)delegate
```

Parameters

delegate
Object to be the receiver's delegate.

Availability
Available in Mac OS X v10.3 and later.

See Also
- delegate (page 2492)

Declared In
NSSpeechSynthesizer.h

setObject:forProperty:error:

Specifies the value of a receiver's property.

```
- (BOOL)setObject:(id)object forProperty:(NSString *)property error:(NSError
    **)outError
```

Parameters

speechProperty
Property to set. The supported properties are listed in "NSSpeechStatusProperty Dictionary Keys" (page 2509).

out_error
On output, error that occurred while setting *speechProperty*.

Return Value
YES when the *speechProperty* was set. NO when there was an error, specified in *out_error*.

Availability
Available in Mac OS X v10.5 and later.

See Also
- objectForProperty:error: (page 2494)

Declared In
NSSpeechSynthesizer.h

setRate:

Specifies the receivers speaking rate.

```
- (void)setRate:(float)rate
```

Parameters

rate

Words to speak per minute.

Availability
Available in Mac OS X v10.5 and later.

See Also
- rate (page 2495)

Declared In
NSSpeechSynthesizer.h

setUsesFeedbackWindow:

Specifies whether the receiver uses the speech feedback window.

```
- (void)setUsesFeedbackWindow:(BOOL)useFeedbackWindow
```

Parameters

useFeedbackWindow

YES to make the receiver use the speech feedback window if it's visible when the user begins speaking. NO not to use the feedback window.

Discussion
See the class description for details on the UsesFeedbackWindow attribute.

Important: The delegate only receives the speechSynthesizer:didFinishSpeaking: (page 3805) message when speaking occurs through the feedback window.

Availability
Available in Mac OS X v10.3 and later.

See Also
- usesFeedbackWindow (page 2501)

Declared In
NSSpeechSynthesizer.h

setVoice:

Sets the receiver's current voice.

```
- (BOOL)setVoice:(NSString *)voiceIdentifier
```

Parameters

voiceIdentifier

Identifier of the voice to set at the receiver's current voice. When nil, the receiver sets the default voice as its current voice.

Return Value
`YES` when the receiver is not currently synthesizing speech and the current voice is set successfully, `NO` otherwise.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- voice` (page 2501)
`+ defaultVoice` (page 2491)

Declared In
`NSSpeechSynthesizer.h`

setVolume:

Specifies the receiver's speaking volume.

```
- (void)setVolume:(float)volume
```

Parameters
volume
Sound level to use for speech.

Discussion
Volumes are expressed in floating-point units ranging from 0.0 through 1.0. A value of 0.0 corresponds to silence, and a value of 1.0 corresponds to the maximum possible volume. Volume units lie on a scale that is linear with amplitude or voltage. A doubling of perceived loudness corresponds to a doubling of the volume.

Setting a value outside this range is undefined.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- volume` (page 2501)

Declared In
`NSSpeechSynthesizer.h`

startSpeakingString:

Begins speaking synthesized text through the system's default sound output device.

```
- (BOOL)startSpeakingString:(NSString *)text
```

Parameters
text
Text to speak. When `nil` or empty, no synthesis occurs.

Return Value
`YES` when speaking starts successfully, `NO` otherwise.

Discussion
If the receiver is currently speaking synthesized speech when `startSpeakingString:` is called, that process is stopped before *text* is spoken.

When synthesis of *text* finishes normally or is stopped, the message `speechSynthesizer:didFinishSpeaking:` (page 3805) is sent to the delegate.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `isSpeaking` (page 2493)
- `startSpeakingString:toURL:` (page 2499)
- `stopSpeaking` (page 2500)

Related Sample Code
SayIt

Declared In
`NSSpeechSynthesizer.h`

startSpeakingString:toURL:

Begins synthesizing text into a sound (AIFF) file.

```
- (BOOL)startSpeakingString:(NSString *)text  toURL:(NSURL *)url
```

Parameters

text
Text to speak. When `nil` or empty, no synthesis is started.

url
Filesystem location of the output sound file.

Return Value
`YES` when synthesis starts successfully, `NO` otherwise.

Discussion
When synthesis of *text* finishes normally or is stopped, the message `speechSynthesizer:didFinishSpeaking:` (page 3805) is sent to the delegate.

One example of how you might use this method is in an email program that automatically converts new messages into sound files that can be stored on an iPod for later listening.

Note: In Mac OS X V 10.4 and earlier, the delegate does not receive `speechSynthesizer:willSpeakWord:ofString:` (page 3806) and `speechSynthesizer:willSpeakPhoneme:` (page 3805) messages when text is being synthesized to a file.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `isSpeaking` (page 2493)

- startSpeakingString: (page 2498)
- stopSpeaking (page 2500)

Declared In
NSSpeechSynthesizer.h

stopSpeaking

Stops synthesis in progress.

- (void)stopSpeaking

Discussion
If the receiver is currently generating speech, synthesis is halted, and the message speechSynthesizer:didFinishSpeaking: (page 3805) is sent to the delegate.

Availability
Available in Mac OS X v10.3 and later.

See Also
- isSpeaking (page 2493)
- startSpeakingString: (page 2498)
- startSpeakingString:toURL: (page 2499)

Related Sample Code
SayIt

Declared In
NSSpeechSynthesizer.h

stopSpeakingAtBoundary:

Stops synthesis in progress at a given boundary.

- (void)stopSpeakingAtBoundary:(NSSpeechBoundary)*boundary*

Parameters
boundary
Boundary at which to stop speech. The supported bound types are listed in "NSSpeechBoundary" (page 2514).

Discussion
Pass the constant NSSpeechImmediateBoundary (page 2514) to stop immediately, even in the middle of a word. Pass NSSpeechWordBoundary (page 2514) or NSSpeechSentenceBoundary (page 2514) to stop speech at the end of the current word or sentence, respectively.

If the end of the string being spoken is reached before the specified stopping point, the synthesizer stops at the end of the string without generating an error.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSSpeechSynthesizer.h

usesFeedbackWindow

Indicates whether the receiver uses the speech feedback window.

```
- (BOOL)usesFeedbackWindow
```

Return Value
YES when the receiver uses the speech feedback window, NO otherwise.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setUsesFeedbackWindow: (page 2497)

Declared In
NSSpeechSynthesizer.h

voice

Returns the identifier of the receiver's current voice.

```
- (NSString *)voice
```

Return Value
Identifier of the receiver's current voice.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setVoice: (page 2497)

Declared In
NSSpeechSynthesizer.h

volume

Provides the receiver's speaking volume.

```
- (float)volume
```

Return Value
Speaking volume: From 0.0 (minimum) to 1.0 (maximum).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setVolume:` (page 2498)

Declared In
`NSSpeechSynthesizer.h`

Constants

Voice Attributes Keys

The following constants are keys for the dictionary returned by `attributesForVoice:` (page 2490).

```
NSString *const NSVoiceIdentifier;
NSString *const NSVoiceName;
NSString *const NSVoiceAge;
NSString *const NSVoiceGender;
NSString *const NSVoiceDemoText;
NSString *const NSVoiceLanguage;
NSString *const NSVoiceLocaleIdentifier;
NSString *const NSVoiceSupportedCharacters;
NSString *const NSVoiceIndividuallySpokenCharacters;
```

Constants

`NSVoiceIdentifier`
A unique string identifying the voice. The identifiers of the system voices are listed in Listing 127-1 (page 2503).
Available in Mac OS X v10.3 and later.
Declared in `NSSpeechSynthesizer.h`.

`NSVoiceName`
The name of the voice suitable for display. An `NSString`.
Available in Mac OS X v10.3 and later.
Declared in `NSSpeechSynthesizer.h`.

`NSVoiceAge`
The perceived age (in years) of the voice. An `NSString`
Available in Mac OS X v10.3 and later.
Declared in `NSSpeechSynthesizer.h`.

`NSVoiceGender`
The perceived gender of the voice. The supported values are listed in "Voice Genders" (page 2504). An `NSString`
Available in Mac OS X v10.3 and later.
Declared in `NSSpeechSynthesizer.h`.

`NSVoiceDemoText`
A demonstration string to speak. An `NSString`
Available in Mac OS X v10.3 and later.
Declared in `NSSpeechSynthesizer.h`.

`NSVoiceLanguage`

The language of the voice (currently US English only). An `NSString`

Deprecated: Use `NSVoiceLocaleIdentifier` (page 2503) instead.

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.5.

Declared in `NSSpeechSynthesizer.h`.

`NSVoiceLocaleIdentifier`

The language of the voice. An `NSString`

The canonical locale identifier string describing the voice's locale. A locale is generally composed of three pieces of ordered information: a language code, a region code, and a variant code. Refer to documentation about the `NSLocale` class or *Locales Programming Guide* for more information.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSVoiceSupportedCharacters`

A list of unicode character id ranges that define the unicode characters supported by this voice. a dictionary containing two keys: "UnicodeCharBegin", an integer value containing the beginning unicode id of this range; and "UnicodeCharBegin", an integer value containing the ending unicode id of this range. The synthesizer will convert or ignore any characters not contained in the range of supported characters.

Some voices may not provide this attribute.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSVoiceIndividuallySpokenCharacters`

A list of unicode character id ranges that define the unicode characters that can be spoken in character-by-character mode by this voice. ach list entry is a dictionary containing two keys: "UnicodeCharBegin", an integer value containing the beginning unicode id of this range; and "UnicodeCharBegin", an integer value containing the ending unicode id of this range.

These ranges can be used by your application to determine if the voice can speak the name of an individual character when spoken in character-by-character mode.

Ssome voices may not provide this attribute.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

Discussion

Listing 127-1 lists the identifiers of the system voices (defined in `/System/Library/Speech/Voices`):

Listing 127-1 Identifiers of the Mac OS X system voices

```
com.apple.speech.synthesis.voice.Agnes
com.apple.speech.synthesis.voice.Albert
com.apple.speech.synthesis.voice.Alex
com.apple.speech.synthesis.voice.BadNews
com.apple.speech.synthesis.voice.Bahh
com.apple.speech.synthesis.voice.Bells
com.apple.speech.synthesis.voice.Boing
com.apple.speech.synthesis.voice.Bruce
com.apple.speech.synthesis.voice.Bubbles
com.apple.speech.synthesis.voice.Cellos
com.apple.speech.synthesis.voice.Deranged
```

```
com.apple.speech.synthesis.voice.Fred
com.apple.speech.synthesis.voice.GoodNews
com.apple.speech.synthesis.voice.Hysterical
com.apple.speech.synthesis.voice.Junior
com.apple.speech.synthesis.voice.Kathy
com.apple.speech.synthesis.voice.Organ
com.apple.speech.synthesis.voice.Princess
com.apple.speech.synthesis.voice.Ralph
com.apple.speech.synthesis.voice.Trinoids
com.apple.speech.synthesis.voice.Vicki
com.apple.speech.synthesis.voice.Victoria
com.apple.speech.synthesis.voice.Whisper
com.apple.speech.synthesis.voice.Zarvox
```

Voice Gender Keys

The following constants define voice gender attributes, which are the allowable values of the `NSVoiceGender` (page 2502) key returned by `attributesForVoice:` (page 2490).

```
NSString *const NSVoiceGenderNeuter;
NSString *const NSVoiceGenderMale;
NSString *const NSVoiceGenderFemale;
```

Constants

`NSVoiceGenderNeuter`

A neutral voice (or a novelty voice with a humorous or whimsical quality).

Available in Mac OS X v10.3 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSVoiceGenderMale`

A male voice

Available in Mac OS X v10.3 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSVoiceGenderFemale`

A female voice

Available in Mac OS X v10.3 and later.

Declared in `NSSpeechSynthesizer.h`.

Speech Synthesizer Property Keys

These constants are used with `setObject:forProperty:error:` (page 2496) and `objectForProperty:error:` (page 2494) to get or set the characteristics of a synthesizer.

```
NSString *const NSSpeechStatusProperty;
NSString *const NSSpeechErrorsProperty;
NSString *const NSSpeechInputModeProperty;
NSString *const NSSpeechCharacterModeProperty;
NSString *const NSSpeechNumberModeProperty;
NSString *const NSSpeechRateProperty;
NSString *const NSSpeechPitchBaseProperty;
NSString *const NSSpeechPitchModProperty;
NSString *const NSSpeechVolumeProperty;
NSString *const NSSpeechSynthesizerInfoProperty;
NSString *const NSSpeechRecentSyncProperty;
NSString *const NSSpeechPhonemeSymbolsProperty;
NSString *const NSSpeechCurrentVoiceProperty;
NSString *const NSSpeechCommandDelimiterProperty;
NSString *const NSSpeechResetProperty;
NSString *const NSSpeechOutputToFileURLProperty;
```

Constants

`NSSpeechStatusProperty`

Get speech-status information for the synthesizer. An `NSDictionary` that contains speech-status information for the synthesizer. See "NSSpeechStatusProperty Dictionary Keys" (page 2509) for a description of the keys present in the dictionary.

This property is used with `setObject:forProperty:error:` (page 2496).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechErrorsProperty`

Get speech-error information for the synthesizer. An `NSDictionary` object that contains speech-error information. See "NSSpeechErrorProperty Dictionary Keys" (page 2510) for a description of the keys present in the dictionary.

This property lets you get information about various run-time errors that occur during speaking, such as the detection of badly formed embedded commands. Errors returned directly by the Speech Synthesis Manager are not reported here.

If your application implements the `speechSynthesizer:didEncounterErrorAtIndex:ofString:message:` (page 3804) delegate message, the delegate message can use this property to get error information.

This property is used with `setObject:forProperty:error:` (page 2496).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechInputModeProperty`

Get or set the synthesizer's current text-processing mode. An `NSString` object that specifies whether the channel is currently in text input mode or phoneme input mode.The supported values are listed in "Speaking Modes for NSSpeechInputModeProperty" (page 2509).

When in phoneme-processing mode, a text string is interpreted to be a series of characters representing various phonemes and prosodic controls. Some synthesizers might support additional input-processing modes and define constants for these modes.

This property is used with `setObject:forProperty:error:` (page 2496) and `objectForProperty:error:` (page 2494).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechCharacterModeProperty`

Get or set the synthesizer's current text-processing mode. An `NSString` object that specifies whether the channel is currently in text input mode or phoneme input mode.The supported values are listed in "Speaking Modes for NSSpeechInputModeProperty" (page 2509).

When the character-processing mode is `NSSpeechModeNormal` (page 2509), input characters are spoken as you would expect to hear them. When the mode is `NSSpeechModeLiteral` (page 2509), each character is spoken literally, so that the word "cat" is spoken "`C-A-T`".

This property is used with `setObject:forProperty:error:` (page 2496) and `objectForProperty:error:` (page 2494).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechNumberModeProperty`

Get or set the synthesizer's current number-processing mode. An `NSString` object that specifies whether the synthesizer is currently in normal or literal number-processing mode. The constants `NSSpeechModeNormal` (page 2509) and `NSSpeechModeLiteral` (page 2509) are the possible values of this string.

When the number-processing mode is `NSSpeechModeNormal` (page 2509), the synthesizer assembles digits into numbers (so that "12" is spoken as "twelve"). When the mode is `NSSpeechModeLiteral` (page 2509), each digit is spoken literally (so that "12" is spoken as "one, two").

This property is used with `setObject:forProperty:error:` (page 2496) and `objectForProperty:error:` (page 2494).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechRateProperty`

Get or set the synthesizer's baseline speech pitch. An `NSNumber` object that specifies the synthesizer's baseline speech pitch.

Typical voice frequencies range from around 90 hertz for a low-pitched male voice to perhaps 300 hertz for a high-pitched child's voice. These frequencies correspond to approximate pitch values in the ranges of 30.000 to 40.000 and 55.000 to 65.000, respectively.

This property is used with `setObject:forProperty:error:` (page 2496) and `objectForProperty:error:` (page 2494).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechPitchBaseProperty`

Get or set a synthesizer's baseline speech pitch. An `NSNumber`object that specifies the baseline speech pitch.

Typical voice frequencies range from around 90 hertz for a low-pitched male voice to perhaps 300 hertz for a high-pitched child's voice. These frequencies correspond to approximate pitch values in the ranges of 30.000 to 40.000 and 55.000 to 65.000, respectively.

This property is used with `setObject:forProperty:error:` (page 2496) and `objectForProperty:error:` (page 2494).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechPitchModProperty`

Get or set a synthesizer's pitch modulation. An `NSNumber` object that specifies the synthesizer's pitch modulation.

Pitch modulation is also expressed as a floating-point value in the range of 0.000 to 127.000. These values correspond to MIDI note values, where 60.000 is equal to middle C on a piano scale. The most useful speech pitches fall in the range of 40.000 to 55.000. A pitch modulation value of 0.000 corresponds to a monotone in which all speech is generated at the frequency corresponding to the speech pitch. Given a speech pitch value of 46.000, a pitch modulation of 2.000 would mean that the widest possible range of pitches corresponding to the actual frequency of generated text would be 44.000 to 48.000.

This property is used with `setObject:forProperty:error:` (page 2496) and `objectForProperty:error:` (page 2494).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechVolumeProperty`

Get or set the speech volume for a synthesizer. An `NSNumber` that specifies the synthesizer's speech volume.

Volumes are expressed in floating-point values ranging from 0.0 through 1.0. A value of 0.0 corresponds to silence, and a value of 1.0 corresponds to the maximum possible volume. Volume units lie on a scale that is linear with amplitude or voltage. A doubling of perceived loudness corresponds to a doubling of the volume.

This property is used with `setObject:forProperty:error:` (page 2496) and `objectForProperty:error:` (page 2494).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechSynthesizerInfoProperty`

Get information about the speech synthesizer being used on the specified synthesizer. An `NSDictionary` object that contains information about the speech synthesizer being used on the specified synthesizer. See "Speech Synthesizer Property Keys" (page 2504) for a description of the keys present in the dictionary.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechRecentSyncProperty`

Get the message code for the most recently encountered synchronization command. An `NSNumber` object that specifies the most recently encountered synchronization command.

This property works with `setObject:forProperty:error:` (page 2496).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechPhonemeSymbolsProperty`

Get a list of phoneme symbols and example words defined for the synthesizer. An `NSDictionary` object that contains the phoneme symbols and example words defined for the current synthesizer

Your application might use this information to show the user what symbols to use when entering phonemic text directly. See "NSSpeechPhonemeSymbolsProperty Dictionary Keys" (page 2512) for a description of the keys present in the dictionary.

This property works with `setObject:forProperty:error:` (page 2496).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechCurrentVoiceProperty`

Set the current voice on the synthesizer to the specified voice. An `NSDictionary` object that contains the phoneme symbols and example words defined for the current synthesizer.

Your application might use this information to show the user what symbols to use when entering phonemic text directly. See "NSSpeechPhonemeSymbolsProperty Dictionary Keys" (page 2512) for the keys you can use to specify values in this dictionary.

This property works with `setObject:forProperty:error:` (page 2496).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechCommandDelimiterProperty`

Set the embedded speech command delimiter characters to be used for the synthesizer. An `NSDictionary` object that contains the delimiter information. See "Command Delimiter Keys" for the keys you can use to specify values in this dictionary.

By default, the opening delimiter is "[[" and the closing delimiter is "]]". Your application might need to change these delimiters temporarily if those character sequences occur naturally in a text buffer that is to be spoken. Your application can also disable embedded command processing by passing empty delimiters (as empty strings). See "Speech Command Delimiter" (page 2513) for the keys you can use to specify values in this dictionary.

This property works with `setObject:forProperty:error:` (page 2496).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechResetProperty`

Set a synthesizer back to its default state. There is no value associated with this property; to reset the channel to its default state, set the key to `NULL`.

You can use this function to, for example, set speech pitch and speech rate to default values.

This property works with `setObject:forProperty:error:` (page 2496).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechOutputToFileURLProperty`

Set the speech output destination to a file or to the computer's speakers. An `NSURL` object. To write the speech output to a file, use the file's NSURL; to generate the sound through the computer's speakers, use `NULL`.

This property works with `setObject:forProperty:error:` (page 2496).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

Speaking Modes for NSSpeechInputModeProperty

These constants identify input modes are used with `NSSpeechInputModeProperty` (page 2505).

```
NSString *const NSSpeechModeText;
NSString *const NSSpeechModePhoneme;
```

Constants

`NSSpeechModeText`

Indicates that the synthesizer is in text-processing mode.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechModePhoneme`

Indicates that the synthesizer is in phoneme-processing mode. When in phoneme-processing mode, a text buffer is interpreted to be a series of characters representing various phonemes and prosodic controls.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

Speaking Modes for NSSpeechNumberModeProperty

These constants define the available text-processing and number-processing modes for a synthesizer. These keys are used with `NSSpeechInputModeProperty` (page 2505) and `NSSpeechNumberModeProperty` (page 2506))

```
NSString *const NSSpeechModeNormal;
NSString *const NSSpeechModeLiteral;
```

Constants

`NSSpeechModeNormal`

Indicates that the synthesizer assembles digits into numbers (so that 12 is spoken as "twelve") and text into words.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechModeLiteral`

Indicates that each digit or character is spoken literally (so that 12 is spoken as "one, two", or the word "cat" is spoken as "C A T").

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

NSSpeechStatusProperty Dictionary Keys

These constants identify speech status keys used with `NSSpeechStatusProperty` (page 2505).

```
NSString *const NSSpeechStatusOutputBusy;
NSString *const NSSpeechStatusOutputPaused;
NSString *const NSSpeechStatusNumberOfCharactersLeft;
NSString *const NSSpeechStatusPhonemeCode;
```

Constants

`NSSpeechStatusOutputBusy`

Indicates whether the synthesizer is currently producing speech.

A synthesizer is considered to be producing speech even at some times when no audio data is being produced through the computer's speaker. This occurs, for example, when the synthesizer is processing input, but has not yet initiated speech or when speech output is paused.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechStatusOutputPaused`

Indicates whether speech output in the synthesizer has been paused by sending the message `pauseSpeakingAtBoundary:` (page 2494).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechStatusNumberOfCharactersLeft`

The number of characters left in the input string of text.

When the value of this key is zero, you can destroy the input string.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechStatusPhonemeCode`

Indicates that the synthesizer is in phoneme-processing mode. When in phoneme-processing mode, a text buffer is interpreted to be a series of characters representing various phonemes and prosodic controls.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

NSSpeechErrorProperty Dictionary Keys

These key constants identify errors that may occur during speech synthesis. They are used with `NSSpeechErrorsProperty` (page 2505).

```
NSString *const NSSpeechErrorCount;
NSString *const NSSpeechErrorOldestCode;
NSString *const NSSpeechErrorOldestCharacterOffset;
NSString *const NSSpeechErrorNewestCode;
NSString *const NSSpeechErrorNewestCharacterOffset;
```

Constants

`NSSpeechErrorCount`

The number of errors that have occurred in processing the current text string, since the last call to `objectForProperty:error:` (page 2494) with the `NSSpeechErrorsProperty` (page 2505) property. An `NSNumber`

Using the `NSSpeechErrorOldestCode` (page 2511) keys and the `NSSpeechErrorNewestCode` (page 2511) keys, you can get information about the oldest and most recent errors that occurred since the last call to `objectForProperty:error:` (page 2494), but you cannot get information about any intervening errors.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechErrorOldestCode`

The error code of the first error that occurred since the last call to `objectForProperty:error:` (page 2494) with the `NSSpeechErrorsProperty` (page 2505) property. An `NSNumber`

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechErrorOldestCharacterOffset`

The position in the text string of the first error that occurred since the last call to `objectForProperty:error:` (page 2494) with the `NSSpeechErrorsProperty` (page 2505) property. An `NSNumber`

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechErrorNewestCode`

The error code of the most recent error that occurred since the last call to `objectForProperty:error:` (page 2494) with the `NSSpeechErrorsProperty` (page 2505) property. An `NSNumber`

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechErrorNewestCharacterOffset`

The position in the text string of the most recent error that occurred since the last call to `objectForProperty:error:` (page 2494) with the `NSSpeechErrorsProperty` (page 2505) property. An `NSNumber`.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

NSSpeechSynthesizerInfoProperty Dictionary Keys

These constants are keys used in the NSSpeechSynthesizerInfoProperty dictionary.

```
NSString *const NSSpeechSynthesizerInfoIdentifier;
NSString *const NSSpeechSynthesizerInfoVersion;
```

Constants

`NSSpeechSynthesizerInfoIdentifier`

The identifier of the speech synthesizer.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechSynthesizerInfoVersion`

The version of the speech synthesizer.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

NSSpeechPhonemeSymbolsProperty Dictionary Keys

These constants are keys used in the NSSpeechPhonemeSymbolsProperty dictionary.

```
NSString *const NSSpeechPhonemeInfoOpcode
NSString *const NSSpeechPhonemeInfoSymbol;
NSString *const NSSpeechPhonemeInfoExample;
NSString *const NSSpeechPhonemeInfoHiliteStart;
NSString *const NSSpeechPhonemeInfoHiliteEnd;
```

Constants

`NSSpeechPhonemeInfoOpcode`

NSNumber

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechPhonemeInfoSymbol`

The symbol used to represent the phoneme.

The symbol does not necessarily have a phonetic connection to the phoneme, but might simply be an abstract textual representation of it.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechPhonemeInfoExample`

An example word that illustrates the use of the phoneme.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechPhonemeInfoHiliteStart`

The character offset into the example word that identifies the location of the beginning of the phoneme.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechPhonemeInfoHiliteEnd`

The character offset into the example word that identifies the location of the end of the phoneme.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

Speech Command Delimiter Keys

These constants speech-command delimiters keys used in `NSSpeechCommandDelimiterProperty` (page 2508).

```
NSString *const NSSpeechCommandPrefix;
NSString *const NSSpeechCommandSuffix;
```

Constants

`NSSpeechCommandPrefix`

The command delimiter string that prefixes a command, by default, this is `[[`.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechCommandSuffix`

The command delimiter string that suffixes a command,by default, this is `]]`.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

Speech Dictionary Properties Keys

These constants identify key-value pairs used to add vocabulary to the dictionary using `addSpeechDictionary:` (page 2491).

```
NSString *const NSSpeechDictionaryLocaleIdentifier;
NSString *const NSSpeechDictionaryModificationDate;
NSString *const NSSpeechDictionaryPronunciations;
NSString *const NSSpeechDictionaryAbbreviations;
NSString *const NSSpeechDictionaryEntrySpelling;
NSString *const NSSpeechDictionaryEntryPhonemes;
```

Constants

`NSSpeechDictionaryLocaleIdentifier`

The canonical locale identifier string describing the dictionary's locale. A locale is generally composed of three pieces of ordered information: a language code, a region code, and a variant code. Refer to documentation about `NSLocale` or *Locales Programming Guide* for more information

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechDictionaryModificationDate`

A string representation of the dictionary's last modification date in the international format (YYYY-MM-DD HH:MM:SS ±HHMM). If the same word appears across multiple dictionaries, the one from the dictionary with the most recent date will be used.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechDictionaryPronunciations`

An array of dictionary objects containing the keys `NSSpeechDictionaryEntrySpelling` (page 2514) and `NSSpeechDictionaryEntryPhonemes` (page 2514).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechDictionaryAbbreviations`
An array of dictionary objects containing the keys `NSSpeechDictionaryEntrySpelling` (page 2514) and `NSSpeechDictionaryEntryPhonemes` (page 2514).

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechDictionaryEntrySpelling`
The spelling of an entry. An `NSString`.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechDictionaryEntryPhonemes`
The phonemic representation of an entry. An `NSString`.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

NSSpeechBoundary

These constants are used to indicate where speech should be stopped and paused. See `pauseSpeakingAtBoundary:` (page 2494) and `stopSpeakingAtBoundary:` (page 2500).

```
enum {
    NSSpeechImmediateBoundary =  0,
    NSSpeechWordBoundary,
    NSSpeechSentenceBoundary
};
typedef NSUInteger NSSpeechBoundary;
```

Constants

`NSSpeechImmediateBoundary`
Speech should be paused or stopped immediately.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechWordBoundary`
Speech should be paused or stopped at the end of the word.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

`NSSpeechSentenceBoundary`
Speech should be paused or stopped at the end of the sentence.

Available in Mac OS X v10.5 and later.

Declared in `NSSpeechSynthesizer.h`.

NSSpellChecker Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSSpellChecker.h
Companion guide	Spell Checking
Related sample code	SpellingChecker CarbonCocoa Bundled SpellingChecker-CarbonCocoa SpellingChecker-CocoaCarbon

Overview

The `NSSpellChecker` class provides an interface to the Cocoa spell-checking service. To handle all its spell checking, an application needs only one instance of `NSSpellChecker`, known as the **spell checker**. Through the spell checker you manage the **Spelling panel**, in which the user can specify decisions about words that are suspect.

The spell checker also offers the ability to provide word completions to augment the text completion system in Mac OS X v10.3.

Tasks

Getting the Spell Checker

`+ sharedSpellChecker` (page 2518)
Returns the NSSpellChecker (one per application).

`+ sharedSpellCheckerExists` (page 2518)
Returns whether the application's NSSpellChecker has already been created.

Configuring Spell Checkers Languages

- `availableLanguages` (page 2520)
  Provides a list of all available languages.
- `userPreferredLanguages` (page 2535)
  Provides a subset of the available languages to be used for spell checking.
- `automaticallyIdentifiesLanguages` (page 2520)
  Returns whether the spell checker will automatically identify languages.
- `setAutomaticallyIdentifiesLanguages:` (page 2531)
  Sets whether the spell checker will automatically identify languages.
- `language` (page 2528)
  Returns the current language used in spell checking.
- `setLanguage:` (page 2531)
  Returns whether the specified language is in the Spelling pop-up list.

Managing Panels

- `spellingPanel` (page 2533)
  Returns the spell checker's panel.
- `substitutionsPanel` (page 2533)
  Returns the substitutions panel.
- `updateSpellingPanelWithGrammarString:detail:` (page 2535)
  Specifies a grammar-analysis detail to highlight in the Spelling panel.
- `updatePanels` (page 2534)
  Updates the available panels to account for user changes.
- `substitutionsPanelAccessoryViewController` (page 2533)
  Returns the substitutions panel's accessory view controller.
- `setSubstitutionsPanelAccessoryViewController:` (page 2532)
  Sets the substitutions panel's accessory view.
- `setAccessoryView:` (page 2530) Deprecated in Mac OS X v10.5
  Makes an view an accessory of the Spelling panel by making it a subview of the panel's content view.
- `accessoryView` (page 2519) Deprecated in Mac OS X v10.5 Deprecated in Mac OS X v10.5 Deprecated in Mac OS X v10.5
  Returns the Spelling panel's accessory view.

Checking Strings for Spelling and Grammar

- `countWordsInString:language:` (page 2524)
  Returns the number of words in *`stringToCount`*.
- `checkSpellingOfString:startingAt:` (page 2521)
  Starts the search for a misspelled word in *`stringToCheck`* starting at *`startingOffset`* within the string object.

- checkSpellingOfString:startingAt:language:wrap:inSpellDocumentWithTag:wordCount: (page 2522)
  Starts the search for a misspelled word in a string starting at specified offset within the string.
- checkGrammarOfString:startingAt:language:wrap:inSpellDocumentWithTag:details: (page 2520)
  Initiates a grammatical analysis of a given string.
- checkString:range:types:options:inSpellDocumentWithTag:orthography:wordCount: (page 2523)
  Requests unified text checking for the given range of the given string.
- requestCheckingOfString:range:types:options:inSpellDocumentWithTag:completionHandler: (page 2529)
  Requests that the string be checked in the background.
- guessesForWordRange:inString:language:inSpellDocumentWithTag: (page 2526)
  Returns an array of possible substitutions for the specified string.
- guessesForWord: (page 2525) Deprecated in Mac OS X v10.6
  Returns an array of suggested spellings for the misspelled word. (Deprecated. Use guessesForWordRange:inString:language:inSpellDocumentWithTag: (page 2526) instead.)

Managing the Spell-Checking Process

+ uniqueSpellDocumentTag (page 2519)
  Returns a guaranteed unique tag to use as the spell-document tag for a document.
- closeSpellDocumentWithTag: (page 2523)
  Notifies the receiver that the user has finished with the tagged document.
- ignoreWord:inSpellDocumentWithTag: (page 2527)
  Instructs the spell checker to ignore all future occurrences of *wordToIgnore* in the document identified by *tag*.
- ignoredWordsInSpellDocumentWithTag: (page 2527)
  Returns the array of ignored words for a document identified by *tag*.
- setIgnoredWords:inSpellDocumentWithTag: (page 2531)
  Initializes the ignored-words document (a dictionary identified by *tag* with *someWords*), an array of words to ignore.
- setWordFieldStringValue: (page 2533)
  Sets the string that appears in the misspelled word field, using the string object *aString*.
- updateSpellingPanelWithMisspelledWord: (page 2535)
  Causes the spell checker to update the Spelling panel's misspelled-word field to reflect *word*.
- completionsForPartialWordRange:inString:language:inSpellDocumentWithTag: (page 2524)
  Provides a list of complete words that the user might be trying to type based on a partial word in a given string.
- hasLearnedWord: (page 2526)
  Indicates whether the spell checker has learned a given word.
- unlearnWord: (page 2534)
  Tells the spell checker to unlearn a given word.
- forgetWord: (page 2525)
  Remove this word from the spelling dictionary. (Deprecated. Use unlearnWord: (page 2534) instead.)

- `learnWord:` (page 2528)
  Adds the word to the spell checker dictionary.
- `userQuotesArrayForLanguage:` (page 2536)
  Returns the default values for quote replacement.
- `userReplacementsDictionary` (page 2536)
  Returns the dictionary used when replacing words.

Data Detector Interaction

- `menuForResult:string:options:atLocation:inView:` (page 2528)
  Provides a menu containing contextual menu items suitable for certain kinds of detected results.

Class Methods

sharedSpellChecker

Returns the NSSpellChecker (one per application).

```
+ (NSSpellChecker *)sharedSpellChecker
```

Return Value
The spelling checker shared by this application.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ sharedSpellCheckerExists` (page 2518)

Related Sample Code
SpellingChecker CarbonCocoa Bundled
SpellingChecker-CarbonCocoa
SpellingChecker-CocoaCarbon

Declared In
`NSSpellChecker.h`

sharedSpellCheckerExists

Returns whether the application's NSSpellChecker has already been created.

```
+ (BOOL)sharedSpellCheckerExists
```

Return Value
`YES` if the shared spell checker already exists, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ sharedSpellChecker` (page 2518)

Declared In
`NSSpellChecker.h`

uniqueSpellDocumentTag

Returns a guaranteed unique tag to use as the spell-document tag for a document.

```
+ (NSInteger)uniqueSpellDocumentTag
```

Return Value
Returns a unique tag to identified this spell checked object.

Discussion
Use this method to generate tags to avoid collisions with other objects that can be spell checked.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
SpellingChecker CarbonCocoa Bundled
SpellingChecker-CarbonCocoa
SpellingChecker-CocoaCarbon

Declared In
`NSSpellChecker.h`

Instance Methods

accessoryView

Returns the Spelling panel's accessory view.

```
- (NSView *)accessoryView
```

Return Value
The accessory view or `nil` if there is none.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAccessoryView:` (page 2530)

Declared In
`NSSpellChecker.h`

automaticallyIdentifiesLanguages

Returns whether the spell checker will automatically identify languages.

```
- (BOOL)automaticallyIdentifiesLanguages
```

Return Value
`YES` if languages are automatically identified, otherwise `NO`.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setAutomaticallyIdentifiesLanguages:` (page 2531)

Declared In
`NSSpellChecker.h`

availableLanguages

Provides a list of all available languages.

```
- (NSArray *)availableLanguages
```

Return Value
An array containing all the available spell checking languages. The languages are ordered in the user's preferred order as set in the system preferences.

Discussion
If `automaticallyIdentifiesLanguages` (page 2520) is `YES`, then text checking will automatically use this method as appropriate; otherwise, it will use the language set by `setLanguage:` (page 2531).

The older `checkSpellingOfString:startingAt:language:wrap:inSpellDocumentWithTag:wordCount:` (page 2522) and `checkGrammarOfString:startingAt:language:wrap:inSpellDocumentWithTag:details:` (page 2520). methods will use the language set by `setLanguage:` (page 2531), if they are called with a `nil` language argument.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- userPreferredLanguages` (page 2535)

Declared In
`NSSpellChecker.h`

checkGrammarOfString:startingAt:language:wrap:inSpellDocumentWithTag:details:

Initiates a grammatical analysis of a given string.

```
- (NSRange)checkGrammarOfString:(NSString *)string startingAt:(NSInteger)start
    language:(NSString *)language wrap:(BOOL)wrap
    inSpellDocumentWithTag:(NSInteger)documentTag details:(NSArray **)outDetails
```

Parameters

string

String to analyze.

start

Location within *string* at which to start the analysis.

language

Language use in *string*. When `nil`, the language selected in the Spelling panel is used.

wrap

`YES` to specify that the analysis continue to the beginning of string when the end is reached.

`NO` to have the analysis stop at the end of string.

documentTag

An identifier unique within the application used to inform the spell checker which document that text is associated, potentially for many purposes, not necessarily just for ignored words. A value of 0 can be passed in for text not associated with a particular document.

outDetails

On output, dictionaries describing grammar-analysis details within the flagged grammatical unit. See the `NSSpellServer` class for information about these dictionaries.

Return Value
Location of the first flagged grammatical unit.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSSpellChecker.h`

checkSpellingOfString:startingAt:

Starts the search for a misspelled word in *stringToCheck* starting at *startingOffset* within the string object.

```
- (NSRange)checkSpellingOfString:(NSString *)stringToCheck
    startingAt:(NSInteger)startingOffset
```

Parameters

stringToCheck

The string to spell check.

startingOffset

The offset at which to start checking.

Return Value
Returns the range of the first misspelled word.

Discussion
Wrapping occurs, but no ignored-words dictionary is used.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
SpellingChecker CarbonCocoa Bundled
SpellingChecker-CarbonCocoa
SpellingChecker-CocoaCarbon

Declared In
NSSpellChecker.h

checkSpellingOfString:startingAt:language:wrap:inSpellDocumentWithTag:wordCount:

Starts the search for a misspelled word in a string starting at specified offset within the string.

```
- (NSRange)checkSpellingOfString:(NSString *)stringToCheck
    startingAt:(NSInteger)startingOffset language:(NSString *)language
    wrap:(BOOL)wrapFlag inSpellDocumentWithTag:(NSInteger)tag wordCount:(NSInteger
     *)wordCount
```

Parameters

stringToCheck
The string object containing the words to spellcheck.

startingOffset
The offset within *stringToCheck* at which to begin spellchecking.

language
The language of the words in the string. If *language* is nil, or if you obtain the value by sending language (page 2528) to self, the current selection in the Spelling panel's pop-up menu is used. Do not pass in an empty string for *language*.

wrapFlag
YES to indicate that spell checking should continue at the beginning of the string when the end of the string is reached; NO to indicate that spellchecking should stop at the end of the document.

tag
An identifier unique within the application used to inform the spell checker which document that text is associated, potentially for many purposes, not necessarily just for ignored words. A value of 0 can be passed in for text not associated with a particular document.

wordCount
Returns by indirection a count of the words spell-checked up to and including the first error (if any), or -1 if the spell checker fails or does not support word counting. Specify NULL if you do not want this word count.

Return Value
The range of the first misspelled word and optionally (and by reference) the count of words spellchecked in the string in *wordCount*.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
SpellingChecker CarbonCocoa Bundled
SpellingChecker-CarbonCocoa

SpellingChecker-CocoaCarbon

Declared In
NSSpellChecker.h

checkString:range:types:options:inSpellDocumentWithTag:orthography:wordCount:

Requests unified text checking for the given range of the given string.

```
- (NSArray *)checkString:(NSString *)stringToCheck range:(NSRange)range
    types:(NSTextCheckingTypes)checkingTypes options:(NSDictionary *)options
    inSpellDocumentWithTag:(NSInteger)tag orthography:(NSOrthography **)orthography
     wordCount:(NSInteger *)wordCount
```

Parameters

stringToCheck
The string to check.

range
The range of the string to check.

checkingTypes
The type of checking to be performed. The possible constants are listed in NSTextCheckingType and can be combined using the C bit-wise OR operator to perform multiple checks at the same time.

options
The options dictionary specifying the types of checking to perform. See "Spell Checking Option Dictionary Keys" (page 2537) for the possible keys and expected values.

tag
An identifier unique within the application used to inform the spell checker which document that text is associated, potentially for many purposes, not necessarily just for ignored words. A value of 0 can be passed in for text not associated with a particular document.

orthography
Returns by-reference, the orthography of the range of the string. See NSOrthography for more information.

wordCount
Returns by-reference, the word count for the range of the string.

Return Value
An array of NSTextCheckingResult objects describing particular items found during checking and their individual ranges, sorted by range origin, then range end, then result type.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSSpellChecker.h

closeSpellDocumentWithTag:

Notifies the receiver that the user has finished with the tagged document.

```
- (void)closeSpellDocumentWithTag:(NSInteger)tag
```

Discussion
The spell checker will release any resources associated with the document, including but not necessarily limited to, ignored words.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
SpellingChecker CarbonCocoa Bundled
SpellingChecker-CarbonCocoa
SpellingChecker-CocoaCarbon

Declared In
`NSSpellChecker.h`

completionsForPartialWordRange:inString:language:inSpellDocumentWithTag:

Provides a list of complete words that the user might be trying to type based on a partial word in a given string.

```
- (NSArray *)completionsForPartialWordRange:(NSRange)partialWordRange
    inString:(NSString *)string language:(NSString *)language
    inSpellDocumentWithTag:(NSInteger)spellDocumentTag
```

Parameters

partialWordRange
Range that identifies a partial word in *string*.

string
String with the partial word from which to generate the result.

language
Language to used in *string*. When `nil`, this method uses the language selected in the Spelling panel.

spellDocumentTag
Identifies the spell document with ignored words to use.

Return Value
List of complete words from the spell checker dictionary in the order they should be presented to the user.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSSpellChecker.h`

countWordsInString:language:

Returns the number of words in *stringToCount*.

```
- (NSInteger)countWordsInString:(NSString *)stringToCount language:(NSString
    *)language
```

Parameters

stringToCount

The string to count the words in.

language

The language of the string.

Return Value

The number of words in the string.

Discussion

If *language* is `nil`, the current selection in the Spelling panel's pop-up menu is used.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSSpellChecker.h`

forgetWord:

Remove this word from the spelling dictionary. (Deprecated in Mac OS X v10.5. Use `unlearnWord:` (page 2534) instead.)

```
- (void)forgetWord:(NSString *)word
```

Parameters

word

The word to remove.

Availability

Available in Mac OS X v10.5 and later.

Deprecated in Mac OS X v10.5.

Declared In

`NSSpellChecker.h`

guessesForWord:

Returns an array of suggested spellings for the misspelled word. (Deprecated in Mac OS X v10.6. Use `guessesForWordRange:inString:language:inSpellDocumentWithTag:` (page 2526) instead.)

```
- (NSArray *)guessesForWord:(NSString *)word
```

Parameters

word

A misspelled word.

Return Value

An array of suggested spellings for the word.

Discussion

If *word* contains all capital letters, or its first letter is capitalized, the suggested words are capitalized in the same way.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

Related Sample Code
SpellingChecker CarbonCocoa Bundled
SpellingChecker-CarbonCocoa
SpellingChecker-CocoaCarbon

Declared In
`NSSpellChecker.h`

guessesForWordRange:inString:language:inSpellDocumentWithTag:

Returns an array of possible substitutions for the specified string.

```
- (NSArray *)guessesForWordRange:(NSRange)range inString:(NSString *)string
    language:(NSString *)language inSpellDocumentWithTag:(NSInteger)tag
```

Parameters

range
The range of the string to check.

string
The string to guess

language
The language of the string

tag
An identifier unique within the application used to inform the spell checker which document that text is associated, potentially for many purposes, not necessarily just for ignored words. A value of 0 can be passed in for text not associated with a particular document.

Return Value
An array of strings containing possible replacement words.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSSpellChecker.h`

hasLearnedWord:

Indicates whether the spell checker has learned a given word.

```
- (BOOL)hasLearnedWord:(NSString *)word
```

Parameters

word
Word in question.

Return Value
`YES` when the spell checker has learned word, `NO` otherwise.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `learnWord:` (page 2528)

Declared In
`NSSpellChecker.h`

ignoredWordsInSpellDocumentWithTag:

Returns the array of ignored words for a document identified by *tag*.

```
- (NSArray *)ignoredWordsInSpellDocumentWithTag:(NSInteger)tag
```

Discussion
Invoke this method before `closeSpellDocumentWithTag:` (page 2523) if you want to store the ignored words.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setIgnoredWords:inSpellDocumentWithTag:` (page 2531)

Related Sample Code
SpellingChecker CarbonCocoa Bundled
SpellingChecker-CarbonCocoa
SpellingChecker-CocoaCarbon

Declared In
`NSSpellChecker.h`

ignoreWord:inSpellDocumentWithTag:

Instructs the spell checker to ignore all future occurrences of *wordToIgnore* in the document identified by *tag*.

```
- (void)ignoreWord:(NSString *)wordToIgnore inSpellDocumentWithTag:(NSInteger)tag
```

Discussion
You should invoke this method from within your implementation of the NSIgnoreMisspelledWords protocol's `ignoreSpelling:` (page 3692) method.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
SpellingChecker CarbonCocoa Bundled
SpellingChecker-CarbonCocoa
SpellingChecker-CocoaCarbon

Declared In
NSSpellChecker.h

language

Returns the current language used in spell checking.

```
- (NSString *)language
```

Return Value
The current spell checking language, as a string.

Discussion
The result string specifies the language using the language and regional designations described in Language and Locale Designations in *Internationalization Programming Topics*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setLanguage: (page 2531)

Declared In
NSSpellChecker.h

learnWord:

Adds the word to the spell checker dictionary.

```
- (void)learnWord:(NSString *)word
```

Parameters
word
The word to add.

Availability
Available in Mac OS X v10.5 and later.

See Also
- unlearnWord: (page 2534)

Declared In
NSSpellChecker.h

menuForResult:string:options:atLocation:inView:

Provides a menu containing contextual menu items suitable for certain kinds of detected results.

```
- (NSMenu *)menuForResult:(NSTextCheckingResult *)result string:(NSString
    *)checkedString options:(NSDictionary *)options atLocation:(NSPoint)location
    inView:(NSView *)view
```

Parameters

result

The `NSTextCheckingResult` instance for the checked string.

checkedString

The string that has been checked.

options

The options dictionary allows clients to pass in information associated with the document. See "Spell Checking Option Dictionary Keys" (page 2537) for possible key-value pairs.

location

The location, in the view's coordinate system, to display the menu.

view

The view object over which to display the contextual menu.

Return Value

A menu suitable for displaying as a contextual menu, or adding to another contextual menu as a submenu.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSSpellChecker.h`

requestCheckingOfString:range:types:options:inSpellDocumentWithTag:completionHandler:

Requests that the string be checked in the background.

```
- (NSInteger)requestCheckingOfString:(NSString *)stringToCheck range:(NSRange)range
    types:(NSTextCheckingTypes)checkingTypes options:(NSDictionary *)options
  inSpellDocumentWithTag:(NSInteger)tag completionHandler:(void (^)(NSInteger
  sequenceNumber, NSArray *results, NSOrthography *orthography, NSInteger
  wordCount))completionHandler
```

Parameters

stringToCheck

The string to check.

range

The range of the string to check.

checkingTypes

The type of checking to be performed. The possible constants are listed in `NSTextCheckingType` and can be combined using the C bit-wise `OR` operator to perform multiple checks at the same time.

options

The options dictionary specifying the types of checking to perform. See "Spell Checking Option Dictionary Keys" (page 2537) for the possible keys and expected values.

tag

An identifier unique within the application used to inform the spell checker which document that text is associated, potentially for many purposes, not necessarily just for ignored words. A value of 0 can be passed in for text not associated with a particular document.

completionHandler

The completion handler block object will be called (in an arbitrary context) when results are available, with the sequence number and results.

The block takes four arguments:

sequenceNumber

A monotonically increasing sequence number.

results

An array of `NSTextCheckingResult` objects describing particular items found during checking and their individual ranges, sorted by range origin, then range end, then result type..

orthography

The orthography of the string.

wordCount

The number of words in the range of the string.

Return Value
The return value is a monotonically increasing sequence number that can be used to keep track of requests in flight.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSSpellChecker.h`

setAccessoryView:

Makes an view an accessory of the Spelling panel by making it a subview of the panel's content view.

```
- (void)setAccessoryView:(NSView *)aView
```

Parameters

aView

The accessory view displayed in the receiver.

Discussion
The accessory view can be any custom view you want to display with the spelling panel. The accessory view is displayed below the spelling checker and the panel automatically resizes to accommodate the accessory view.

This method posts a notification named `NSWindowDidResizeNotification` (page 3426) with the Spelling panel object to the default notification center.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- accessoryView` (page 2519)

Declared In
`NSSpellChecker.h`

setAutomaticallyIdentifiesLanguages:

Sets whether the spell checker will automatically identify languages.

```
- (void)setAutomaticallyIdentifiesLanguages:(BOOL)flag
```

Parameters

flag

YES if languages should be automatically identified, otherwise NO.

Availability
Available in Mac OS X v10.6 and later.

See Also
- automaticallyIdentifiesLanguages (page 2520)

Declared In
NSSpellChecker.h

setIgnoredWords:inSpellDocumentWithTag:

Initializes the ignored-words document (a dictionary identified by *tag* with *someWords*), an array of words to ignore.

```
- (void)setIgnoredWords:(NSArray *)someWords inSpellDocumentWithTag:(NSInteger)tag
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- ignoredWordsInSpellDocumentWithTag: (page 2527)

Related Sample Code
SpellingChecker CarbonCocoa Bundled
SpellingChecker-CarbonCocoa
SpellingChecker-CocoaCarbon

Declared In
NSSpellChecker.h

setLanguage:

Returns whether the specified language is in the Spelling pop-up list.

```
- (BOOL)setLanguage:(NSString *)language
```

Parameters

language

The requested language.

Return Value
YES if the language is available in the pop-up list, otherwise NO.

Discussion
Listing 128-1 shows how languages can be specified in language. If the language specified is listed in the user's list of preferred languages, the spell checker uses that language to accomplish its task.

Listing 128-1 Specifying the spell checker language

```
NSSpellChecker* spell_checker = [NSSpellChecker sharedSpellChecker];

// Sets language to French. The language method returns "fr".
[spell_checker setLanguage:@"fr"];

// Sets language to the one spoken in Netherlands (English). The language method
 returns "en".
[spell_checker setLanguage:@"NL"];

// Sets language to British English. The language method returns "en_GB".
[spell_checker setLanguage:@"en_GB"]

 // Sets language to German. The language method returns "de".
[spell_checker setLanguage:@"German"];
```

To learn about the strings you can use to specify a language in *language*, see Language and Locale Designations in *Internationalization Programming Topics*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- language (page 2528)

Declared In
NSSpellChecker.h

setSubstitutionsPanelAccessoryViewController:

Sets the substitutions panel's accessory view.

```
- (void)setSubstitutionsPanelAccessoryViewController:(NSViewController
    *)accessoryController
```

Parameters
accessoryController
The accessory view controller or nil if there is none.

Discussion
The accessory view controller can accommodate be any custom view you want to display with the substitutions panel. The accessory view controller's view is displayed below the substitutions list and the panel automatically resizes to accommodate the accessory view.

This method posts a notification named NSWindowDidResizeNotification (page 3426) with the substitutions panel object to the default notification center.

Availability
Available in Mac OS X v10.6 and later.

See Also
- substitutionsPanelAccessoryViewController (page 2533)

Declared In
NSSpellChecker.h

setWordFieldStringValue:

Sets the string that appears in the misspelled word field, using the string object *aString*.

```
- (void)setWordFieldStringValue:(NSString *)aString
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSSpellChecker.h

spellingPanel

Returns the spell checker's panel.

```
- (NSPanel *)spellingPanel
```

Return Value
The spell checking panel.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSSpellChecker.h

substitutionsPanel

Returns the substitutions panel.

```
- (NSPanel *)substitutionsPanel
```

Return Value
The substitutions checking panel.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSSpellChecker.h

substitutionsPanelAccessoryViewController

Returns the substitutions panel's accessory view controller.

```
- (NSViewController *)substitutionsPanelAccessoryViewController
```

Return Value
The accessory view controller or `nil` if there is none.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setSubstitutionsPanelAccessoryViewController:` (page 2532)

Declared In
`NSSpellChecker.h`

unlearnWord:

Tells the spell checker to unlearn a given word.

```
- (void)unlearnWord:(NSString *)word
```

Parameters
word
Word to unlearn.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- learnWord:` (page 2528)

Declared In
`NSSpellChecker.h`

updatePanels

Updates the available panels to account for user changes.

```
- (void)updatePanels
```

Discussion
This method should be called when a client changes some relevant setting, such as what kind of spelling, grammar checking, or substitutions it uses.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- spellingPanel` (page 2533)
`- substitutionsPanel` (page 2533)

Declared In
`NSSpellChecker.h`

updateSpellingPanelWithGrammarString:detail:

Specifies a grammar-analysis detail to highlight in the Spelling panel.

```
- (void)updateSpellingPanelWithGrammarString:(NSString *)problemString
    detail:(NSDictionary *)detail
```

Parameters

problemString

Problematic grammatical unit identified by `checkGrammarOfString:startingAt:language:wrap:inSpellDocumentWithTag:details:` (page 2520).

detail

One of the grammar-analysis details provided by `checkGrammarOfString:startingAt:language:wrap:inSpellDocumentWithTag:details:` (page 2520).

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSSpellChecker.h`

updateSpellingPanelWithMisspelledWord:

Causes the spell checker to update the Spelling panel's misspelled-word field to reflect *word*.

```
- (void)updateSpellingPanelWithMisspelledWord:(NSString *)word
```

Discussion

You are responsible for highlighting *word* in the document and for extracting it from the document using the range returned by the `checkSpelling:...` methods. Pass the empty string as *word* to have the system beep, indicating no misspelled words were found.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSSpellChecker.h`

userPreferredLanguages

Provides a subset of the available languages to be used for spell checking.

```
- (NSArray *)userPreferredLanguages
```

Return Value

An array containing the user's preferred languages for spell checking. The order is set in the system preferences.

Discussion

If `automaticallyIdentifiesLanguages` (page 2520) is `YES`, then text checking will automatically use this method as appropriate; otherwise, it will use the language set by `setLanguage:` (page 2531).

The older `checkSpellingOfString:startingAt:language:wrap:inSpellDocumentWithTag:wordCount:` (page 2522) and `checkGrammarOfString:startingAt:language:wrap:inSpellDocumentWithTag:details:` (page 2520). methods will use the language set by `setLanguage:` (page 2531), if they are called with a `nil` language argument.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- availableLanguages` (page 2520)

Declared In
`NSSpellChecker.h`

userQuotesArrayForLanguage:

Returns the default values for quote replacement.

```
- (NSArray *)userQuotesArrayForLanguage:(NSString *)language
```

Parameters

language
The language for quote replacement.

Return Value
An array of quote replacements used by the `NSTextCheckingQuotesKey` (page 2537) key-value pair.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- userReplacementsDictionary` (page 2536)

Declared In
`NSSpellChecker.h`

userReplacementsDictionary

Returns the dictionary used when replacing words.

```
- (NSDictionary *)userReplacementsDictionary
```

Return Value
The dictionary.

Discussion
The key-value pairs in this dictionary are used by the `NSTextCheckingQuotesKey` (page 2537) when replacing characters and words.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `userQuotesArrayForLanguage:` (page 2536)

Declared In
`NSSpellChecker.h`

Constants

Spell Checking Option Dictionary Keys

The constants are optional keys that can be used in the options dictionary parameter of the `checkString:range:types:options:inSpellDocumentWithTag:orthography:wordCount:` (page 2523), `requestCheckingOfString:range:types:options:inSpellDocumentWithTag:completionHandler:` (page 2529), and `menuForResult:string:options:atLocation:inView:` (page 2528) methods.

```
NSString *NSTextCheckingOrthographyKey;
NSString *NSTextCheckingQuotesKey;
NSString *NSTextCheckingReplacementsKey;
NSString *NSTextCheckingReferenceDateKey;
NSString *NSTextCheckingReferenceTimeZoneKey;
NSString *NSTextCheckingDocumentURLKey;
NSString *NSTextCheckingDocumentTitleKey;
NSString *NSTextCheckingDocumentAuthorKey;
```

Constants

`NSTextCheckingOrthographyKey`
An `NSOrthography` instance indicating an orthography to be used as a starting point for orthography checking, or as the orthography if orthography checking is not enabled.

Available in Mac OS X v10.6 and later.

Declared in `NSSpellChecker.h`.

`NSTextCheckingQuotesKey`
An NSArray containing four strings to be used with `NSTextCheckingTypeQuote` (opening double quote, closing double quote, opening single quote, and closing single quote in that order); if not specified, values will be taken from user's preferences.

Available in Mac OS X v10.6 and later.

Declared in `NSSpellChecker.h`.

`NSTextCheckingReplacementsKey`
An NSDictionary containing replacements to be used with NSTextCheckingTypeReplacement; if not specified, values will be taken from user's preferences.

Available in Mac OS X v10.6 and later.

Declared in `NSSpellChecker.h`.

`NSTextCheckingReferenceDateKey`
An NSDate to be associated with the document, used as a referent for relative dates; if not specified, the current date will be used.

Available in Mac OS X v10.6 and later.

Declared in `NSSpellChecker.h`.

`NSTextCheckingReferenceTimeZoneKey`

An NSTimeZone to be associated with the document, used as a reference for dates without time zones; if not specified, the current time zone will be used.

Available in Mac OS X v10.6 and later.

Declared in `NSSpellChecker.h`.

`NSTextCheckingDocumentURLKey`

An NSURL to be associated with the document.

Available in Mac OS X v10.6 and later.

Declared in `NSSpellChecker.h`.

`NSTextCheckingDocumentTitleKey`

An NSString containing the title to be associated with the document.

Available in Mac OS X v10.6 and later.

Declared in `NSSpellChecker.h`.

`NSTextCheckingDocumentAuthorKey`

An NSString containing the name of an author to be associated with the document

Available in Mac OS X v10.6 and later.

Declared in `NSSpellChecker.h`.

NSSplitView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSSplitView.h
Companion guides	Scroll View Programming Guide for Cocoa View Programming Guide
Related sample code	ClipboardViewer CoreAnimationText MyPhoto QTMetadataEditor SourceView

Overview

An `NSSplitView` object stacks several subviews within one view so that the user can change their relative sizes. By default, the split bars between the views are horizontal, so the views are one on top of the other.

Divider indices are zero-based, with the topmost (in horizontal split views) or leftmost (vertical) divider having an index of 0.

Tasks

Managing Subviews

`- adjustSubviews` (page 2541)
Adjusts the sizes of the receiver's subviews so they (plus the dividers) fill the receiver.

`- isSubviewCollapsed:` (page 2544)
Returns whether the specified view is collapsed.

Managing Split View Orientation

- `isVertical` (page 2545)
  Returns whether the split bars are vertical.
- `setVertical:` (page 2548)
  Sets whether the split bars are vertical.

Assigning a Delegate

- `delegate` (page 2542)
  Returns the receiver's delegate.
- `setDelegate:` (page 2547)
  Sets the receiver's delegate.

Configuring and Drawing View Dividers

- `setDividerStyle:` (page 2547)
  Sets the style of divider drawn between subviews.
- `dividerStyle` (page 2543)
  Returns the style of the divider drawn between subviews.
- `dividerThickness` (page 2543)
  Returns the thickness of the divider.
- `dividerColor` (page 2542)
  Return the color of the dividers that the split view is drawing between subviews.
- `drawDividerInRect:` (page 2543)
  Draws the divider between two of the receiver's subviews.

Saving Subview Positions

- `setAutosaveName:` (page 2546)
  Sets the name under which receiver's divider position is automatically saved.
- `autosaveName` (page 2541)
  Returns the name under which receiver's divider position is automatically saved.

Configuring Pane Splitters

- `isPaneSplitter` (page 2544) Deprecated in Mac OS X v10.6
  Returns `YES` if the receiver's splitter is a bar that goes across the split view. Returns `NO` if the splitter is a thumb on the regular background pattern. (Deprecated. This functionality is no longer relevant and there is no alternative method.)
- `setIsPaneSplitter:` (page 2547) Deprecated in Mac OS X v10.6
  Sets the type of splitter. (Deprecated. This functionality is no longer relevant and there is no alternative method.)

Constraining Split Position

- `minPossiblePositionOfDividerAtIndex:` (page 2546)
  Returns the minimum possible position of the divider at the specified index.
- `maxPossiblePositionOfDividerAtIndex:` (page 2545)
  Returns the maximum possible position of the divider at the specified index.
- `setPosition:ofDividerAtIndex:` (page 2548)
  Sets the position of the divider at the specified index.

Instance Methods

adjustSubviews

Adjusts the sizes of the receiver's subviews so they (plus the dividers) fill the receiver.

```
- (void)adjustSubviews
```

Discussion
The subviews are resized proportionally; the size of a subview relative to the other subviews doesn't change.

In Mac OS X v10.5 and earlier adjustSubviews did not invalidate the split view's cursor. This made it difficult to correctly animate divider positioning by sending `setFrameSize:` (page 3221) to the `animator` (page 3556) proxy of the subviews on either side of the divider and letting `adjustSubviews` (page 2541) to be invoked repeatedly during the animation. In Mac OS X v10.6 and later `adjustSubviews` (page 2541) now invalidates its cursor so the cursor over the divider is always correct during and after such an animation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDelegate:` (page 2547)
- `setFrame:` (page 3218) (NSView)

Related Sample Code
ClipboardViewer
MyPhoto
PDFKitLinker2

Declared In
`NSSplitView.h`

autosaveName

Returns the name under which receiver's divider position is automatically saved.

```
- (NSString *)autosaveName
```

Return Value
The name used to save the receiver's state.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setAutosaveName:` (page 2546)

Declared In
`NSSplitView.h`

delegate

Returns the receiver's delegate.

```
- (id < NSSplitViewDelegate >)delegate
```

Return Value
The receiver's delegate object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDelegate:` (page 2547)

Declared In
`NSSplitView.h`

dividerColor

Return the color of the dividers that the split view is drawing between subviews.

```
- (NSColor *)dividerColor
```

Return Value
The color of the divider drawn between the subviews.

Discussion
The default implementation of this method returns `clearColor` (page 668) when `dividerStyle` (page 2543) returns `NSSplitViewDividerStyleThick` (page 2549) or when `dividerStyle` (page 2543) returns `NSSplitViewDividerStylePaneSplitter` (page 2549) and the receiver is in a textured window. All other thin dividers are drawn with a color that looks good between two white panes.

You can override this method to change the color of dividers.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSSplitView.h`

dividerStyle

Returns the style of the divider drawn between subviews.

```
- (NSSplitViewDividerStyle)dividerStyle
```

Return Value
The current divider style. The possible values are described in "Split View Divider Styles" (page 2549).

Availability
Available in Mac OS X v10.5 and later.

See Also
- setDividerStyle: (page 2547)

Declared In
NSSplitView.h

dividerThickness

Returns the thickness of the divider.

```
- (CGFloat)dividerThickness
```

Return Value
The thickness of the divider.

Discussion
You can subclass `NSSplitView` and override this method to change the divider's size, if necessary.

Availability
Available in Mac OS X v10.0 and later.

See Also
- drawDividerInRect: (page 2543)

Related Sample Code
SourceView

Declared In
NSSplitView.h

drawDividerInRect:

Draws the divider between two of the receiver's subviews.

```
- (void)drawDividerInRect:(NSRect)aRect
```

Parameters
aRect
The entire divider rectangle in the receiver's coordinate system, which is flipped.

Discussion
If you override this method and use a custom icon to identify the divider, you may need to change the size of the divider.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `dividerThickness` (page 2543)
- `compositeToPoint:operation:` (page 1342) (NSImage)

Declared In
`NSSplitView.h`

isPaneSplitter

Returns `YES` if the receiver's splitter is a bar that goes across the split view. Returns `NO` if the splitter is a thumb on the regular background pattern. (Deprecated in Mac OS X v10.6. This functionality is no longer relevant and there is no alternative method.)

```
- (BOOL)isPaneSplitter
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `setIsPaneSplitter:` (page 2547)

Declared In
`NSSplitView.h`

isSubviewCollapsed:

Returns whether the specified view is collapsed.

```
- (BOOL)isSubviewCollapsed:(NSView *)subview
```

Parameters
subview
The subview in the splitview.

Return Value
`YES` if *subview* is in a collapsed state, `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSplitView.h`

isVertical

Returns whether the split bars are vertical.

```
- (BOOL)isVertical
```

Return Value
`YES` if the split bars are vertical (subviews are side by side), `NO` if they are horizontal (views are one on top of the other).

Discussion
By default, split bars are vertical.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setVertical:` (page 2548)

Declared In
`NSSplitView.h`

maxPossiblePositionOfDividerAtIndex:

Returns the maximum possible position of the divider at the specified index.

```
- (CGFloat)maxPossiblePositionOfDividerAtIndex:(NSInteger)dividerIndex
```

Parameters
dividerIndex
The index of the divider.

Return Value
A CGFloat specifying the maximum possible position of the divider.

Discussion
The position is "possible" in that it is dictated by the bounds of the receiver and the current position of other dividers. "Allowable" positions are those that result from letting the delegate apply constraints to the possible positions.

You can invoke this method to determine the range of values that can be usefully passed to `setPosition:ofDividerAtIndex:` (page 2548).

You can also invoke it from delegate methods like `splitView:constrainSplitPosition:ofSubviewAt:` (page 3813) to implement relatively complex behaviors that depend on the current state of the split view.

The results of invoking this method when `adjustSubviews` (page 2541) has not been invoked, and the subview frames are invalid, is undefined.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSSplitView.h`

minPossiblePositionOfDividerAtIndex:

Returns the minimum possible position of the divider at the specified index.

```
- (CGFloat)minPossiblePositionOfDividerAtIndex:(NSInteger)dividerIndex
```

Parameters

dividerIndex

The index of the divider.

Return Value

A CGFloat specifying the minimum possible position of the divider.

Discussion

The position is "possible" in that it is dictated by the bounds of the receiver and the current position of other dividers. "Allowable" positions are those that result from letting the delegate apply constraints to the possible positions.

You can invoke this method to determine the range of values that can be usefully passed to `setPosition:ofDividerAtIndex:` (page 2548).

You can also invoke it from delegate methods like `splitView:constrainSplitPosition:ofSubviewAt:` (page 3813) to implement relatively complex behaviors that depend on the current state of the split view.

The results of invoking this method when `adjustSubviews` (page 2541) has not been invoked, and the subview frames are invalid, is undefined.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSSplitView.h`

setAutosaveName:

Sets the name under which receiver's divider position is automatically saved.

```
- (void)setAutosaveName:(NSString *)autosaveName
```

Parameters

autosaveName

The name used to save the receiver's state.

Discussion

If this value is `nil` or the string is empty no autosaving is done.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- autosaveName` (page 2541)

Declared In

`NSSplitView.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSSplitViewDelegate >)theDelegate
```

Parameters

theDelegate

The receiver's delegate object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- delegate (page 2542)

Declared In

NSSplitView.h

setDividerStyle:

Sets the style of divider drawn between subviews.

```
- (void)setDividerStyle:(NSSplitViewDividerStyle)dividerStyle
```

Parameters

dividerStyle

The divider style. Possible values are described in "Split View Divider Styles" (page 2549).

Availability

Available in Mac OS X v10.5 and later.

See Also

- dividerStyle (page 2543)

Declared In

NSSplitView.h

setIsPaneSplitter:

Sets the type of splitter. (Deprecated in Mac OS X v10.6. This functionality is no longer relevant and there is no alternative method.)

```
- (void)setIsPaneSplitter:(BOOL)flag
```

Parameters

flag

YES if the receiver's splitter is a bar that goes across the split view. NO if the splitter is a thumb on the regular background pattern.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also
- `isPaneSplitter` (page 2544)

Declared In
`NSSplitView.h`

setPosition:ofDividerAtIndex:

Sets the position of the divider at the specified index.

```
- (void)setPosition:(CGFloat)position ofDividerAtIndex:(NSInteger)dividerIndex
```

Parameters

position
The position of the divider

dividerIndex
The index of the divider.

Discussion
The default implementation of this method behaves as if the user were attempting to drag the divider to the proposed position, so the constraints imposed by the delegate are applied and one of the views adjacent to the divider may be collapsed.

This method is not invoked by `NSSplitView` itself.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
ClipboardViewer

Declared In
`NSSplitView.h`

setVertical:

Sets whether the split bars are vertical.

```
- (void)setVertical:(BOOL)flag
```

Parameters

flag
If `YES`, the split bars are vertical (views are side by side); if `NO`, they're horizontal (views are one on top of the other).

Discussion
Split bars are horizontal by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isVertical` (page 2545)

Declared In
NSSplitView.h

Constants

Split View Divider Styles

These constants specify the possible divider styles used by `dividerStyle` (page 2543) and `setDividerStyle:` (page 2547).

```
enum {
    NSSplitViewDividerStyleThick = 1,
    NSSplitViewDividerStyleThin = 2,
    NSSplitViewDividerStylePaneSplitter = 3,
};
typedef NSInteger NSSplitViewDividerStyle;
```

Constants

`NSSplitViewDividerStyleThick`
A thick style divider is displayed between subviews. This is the default.
Available in Mac OS X v10.5 and later.
Declared in `NSSplitView.h`.

`NSSplitViewDividerStyleThin`
A thin style divider is displayed between subviews.
Available in Mac OS X v10.5 and later.
Declared in `NSSplitView.h`.

`NSSplitViewDividerStylePaneSplitter`
A thick style divider with a 3D appearance is displayed between subviews.
Available in Mac OS X v10.6 and later.
Declared in `NSSplitView.h`.

Notifications

NSSplitView declares and posts the following notifications. In addition, it posts notifications declared by its superclass, NSView. See the `NSView` class specification for more information.

NSSplitViewDidResizeSubviewsNotification

Posted after an NSSplitView changes the sizes of some or all of its subviews. The notification object is the NSSplitView that resized its subviews.

Note: In Mac OS X v10.5 and later if the notification is sent because the user is dragging a divider, the `userInfo` dictionary contains a key `@"NSSplitViewDividerIndex"` containing an `NSNumber`-wrapped `NSInteger` that is the index of the divider being dragged. Earlier versions of Mac OS X do not return a user info dictionary in any situation.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- splitViewDidResizeSubviews:` (page 3817) (NSSplitViewDelegate Protocol)

Declared In
`NSSplitView.h`

NSSplitViewWillResizeSubviewsNotification

Posted before an NSSplitView changes the sizes of some or all of its subviews. The notification object is the NSSplitView object that is about to resize its subviews.

Note: In Mac OS X v10.5 and later if the notification is sent because the user is dragging a divider, the `userInfo` dictionary contains a key `@"NSSplitViewDividerIndex"` containing an `NSInteger`-wrapped `NSNumber` that is the index of the divider being dragged. Earlier versions of Mac OS X do not return a user info dictionary in any situation.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- splitViewWillResizeSubviews:` (page 3817) (NSSplitViewDelegate Protocol)

Declared In
`NSSplitView.h`

NSStatusBar Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSStatusBar.h
Companion guide	Status Bar Programming Topics

Overview

The `NSStatusBar` class defines an object that manages a collection of `NSStatusItem` objects displayed within the system-wide menu bar. A status item can be displayed with text or an icon, can provide a menu and a target-action message when clicked, or can be a fully customized view that you create.

Use status items sparingly and only if the alternatives (such as a Dock menu, preference pane, or status window) are not suitable. Because there is limited space in which to display status items, status items are not guaranteed to be available at all times. For this reason, do not rely on them being available and always provide a user preference for hiding your application's status items to free up space in the menu bar.

Tasks

Getting the System-Wide Instance

`+ systemStatusBar` (page 2552)
Returns the system-wide status bar located in the menu bar.

Managing Status Items

`- statusItemWithLength:` (page 2553)
Returns a newly created status item that has been allotted a specified space within the status bar.

`- removeStatusItem:` (page 2553)
Removes the specified status item from the receiver.

Getting Status-Bar Attributes

- `isVertical` (page 2552)
  Returns whether the receiver has a vertical orientation.
- `thickness` (page 2553)
  Returns the thickness of the status bar.

Class Methods

systemStatusBar

Returns the system-wide status bar located in the menu bar.

```
+ (NSStatusBar *)systemStatusBar
```

Return Value
The shared status bar object.

Discussion
The status bar begins at the right side of the menu bar (to the left of Menu Extras and the menu bar clock) and grows to the left as `NSStatusItem` objects are added to it.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSStatusBar.h`

Instance Methods

isVertical

Returns whether the receiver has a vertical orientation.

```
- (BOOL)isVertical
```

Return Value
`YES` if the receiver has a vertical orientation, otherwise `NO`.

Discussion
The status bar returned by `systemStatusBar` (page 2552) is horizontal, so it always returns `NO`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSStatusBar.h`

removeStatusItem:

Removes the specified status item from the receiver.

```
- (void)removeStatusItem:(NSStatusItem *)item
```

Parameters

item

The `NSStatusItem` object to remove.

Discussion

Status items to the left of the specified one in the status bar shift to the right to reclaim its space.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- statusItemWithLength:` (page 2553)

Declared In

`NSStatusBar.h`

statusItemWithLength:

Returns a newly created status item that has been allotted a specified space within the status bar.

```
- (NSStatusItem *)statusItemWithLength:(CGFloat)length
```

Parameters

length

A constant that specifies whether the status item is of fixed width, or variable width. The valid constants are described in "Status Bar Item Length" (page 2554).

Return Value

An `NSStatusItem` object or `nil` if the item could not be created.

Discussion

The receiver does not retain a reference to the status item, so you need to retain it. Otherwise, the object is removed from the status bar when it is deallocated.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- removeStatusItem:` (page 2553)

Declared In

`NSStatusBar.h`

thickness

Returns the thickness of the status bar.

```
- (CGFloat)thickness
```

Return Value
The thickness of the status bar in pixels.

Discussion
The status bar returned by `systemStatusBar` (page 2552) has a thickness of 22 pixels, the thickness of the menu bar.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSStatusBar.h`

Constants

Status Bar Item Length

The following constants are used by `statusItemWithLength:` (page 2553).

```
#define    NSVariableStatusItemLength    (-1)
#define    NSSquareStatusItemLength      (-2)
```

Constants

`NSSquareStatusItemLength`
Sets the status item length to the status bar thickness.
Available in Mac OS X v10.0 and later.
Declared in `NSStatusBar.h`.

`NSVariableStatusItemLength`
Makes the status item length dynamic, adjusting to the width of its contents.
Available in Mac OS X v10.0 and later.
Declared in `NSStatusBar.h`.

NSStatusItem Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSStatusItem.h
Companion guide	Status Bar Programming Topics

Overview

The `NSStatusItem` class represents the individual elements displayed within an `NSStatusBar` object. Instances are created by the `NSStatusBar` method `statusItemWithLength:` (page 2553), which automatically adds the new status item to the status bar. The appearance and behavior of the status item are then set using the various `NSStatusItem` methods, such as `setTitle:` (page 2565) and `setAction:` (page 2561).

Tasks

Getting the Item's Status Bar

- `statusBar` (page 2566)
  Returns the status bar in which the receiver is displayed.

Configuring the Status Item's Appearance

- `setTitle:` (page 2565)
  Sets the string that is displayed at the receiver's position in the status bar.
- `title` (page 2567)
  Returns the string that is displayed at the receiver's position in the status bar.
- `setAttributedTitle:` (page 2562)
  Sets the attributed string that is displayed at the receiver's position in the status bar.
- `attributedTitle` (page 2558)
  Returns the attributed string that is displayed at the receiver's position in the status bar

- `setImage:` (page 2563)
  Sets the image that is displayed at the receiver's position in the status bar to *image*.
- `image` (page 2559)
  Returns the image that is displayed at the receiver's position in the status bar.
- `setAlternateImage:` (page 2562)
  Sets an alternate image to be displayed when a status bar item is highlighted.
- `alternateImage` (page 2557)
  Returns the alternate image that is displayed when a status bar item is highlighted.
- `setLength:` (page 2564)
  Sets the amount of space in the status bar that should be allocated to the receiver.
- `length` (page 2560)
  Returns the amount of space allocated to the receiver within its status bar.
- `setHighlightMode:` (page 2563)
  Sets whether the receiver is highlighted when it is clicked.
- `highlightMode` (page 2559)
  Returns whether the receiver is highlighted when clicked.
- `setToolTip:` (page 2566)
  Sets the tool tip string that is displayed when the cursor pauses over the receiver.
- `toolTip` (page 2567)
  Returns the tool tip string that is displayed when the cursor pauses over the receiver.

Managing the Status Item's Behavior

- `setEnabled:` (page 2563)
  Sets whether the receiver is enabled to respond to clicks.
- `isEnabled` (page 2559)
  Returns whether the receiver is enabled and responding to clicks.
- `setTarget:` (page 2565)
  Sets the target object to which the receiver's action message is sent when the receiver is clicked.
- `target` (page 2567)
  Returns the target to which the receiver's action message is sent when the user clicks the receiver.
- `setAction:` (page 2561)
  Sets the selector that is sent to the receiver's target when the receiver is clicked.
- `action` (page 2557)
  Returns the selector that is sent to the receiver's target when the user clicks the receiver.
- `setDoubleAction:` (page 2562)
  Sets the selector that is sent to the receiver's target when the receiver is double-clicked.
- `doubleAction` (page 2558)
  Returns the selector that is sent to the receiver's target when the user double-clicks the receiver.
- `sendActionOn:` (page 2561)
  Sets the conditions on which the receiver sends action messages to its target.
- `setMenu:` (page 2564)
  Sets the pull-down menu that is displayed when the receiver is clicked.

- `menu` (page 2560)
  Returns the drop-down menu that is displayed when the receiver is clicked.
- `popUpStatusItemMenu:` (page 2560)
  Displays a menu under a custom status bar item.

Managing a Custom View

- `setView:` (page 2566)
  Sets the custom view that is displayed at the receiver's position in the status bar.
- `view` (page 2567)
  Returns the custom view that is displayed at the receiver's position in the status bar.

Drawing a Status Item

- `drawStatusBarBackgroundInRect:withHighlight:` (page 2558)
  Draws the menu background pattern for a custom status-bar item in regular or highlight pattern.

Instance Methods

action

Returns the selector that is sent to the receiver's target when the user clicks the receiver.

```
- (SEL)action
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAction:` (page 2561)
- `target` (page 2567)

Declared In
`NSStatusItem.h`

alternateImage

Returns the alternate image that is displayed when a status bar item is highlighted.

```
- (NSImage *)alternateImage
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setAlternateImage:` (page 2562)

- image (page 2559)

Declared In
NSStatusItem.h

attributedTitle

Returns the attributed string that is displayed at the receiver's position in the status bar

```
- (NSAttributedString *)attributedTitle
```

Discussion
.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setAttributedTitle: (page 2562)
- setTitle: (page 2565)
- title (page 2567)

Declared In
NSStatusItem.h

doubleAction

Returns the selector that is sent to the receiver's target when the user double-clicks the receiver.

```
- (SEL)doubleAction
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setDoubleAction: (page 2562)
- target (page 2567)

Declared In
NSStatusItem.h

drawStatusBarBackgroundInRect:withHighlight:

Draws the menu background pattern for a custom status-bar item in regular or highlight pattern.

```
- (void)drawStatusBarBackgroundInRect:(NSRect)rect  withHighlight:(BOOL)highlight
```

Parameters
rect
A rectangle defining the area of a custom status-bar item.

highlight
YES to draw the background pattern in the standard highlight pattern, NO to not highlight the pattern..

Discussion
You can use this method to help a custom status-bar item emulate the behavior of a standard item.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setView: (page 2566)

Declared In
NSStatusItem.h

highlightMode

Returns whether the receiver is highlighted when clicked.

```
- (BOOL)highlightMode
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- setHighlightMode: (page 2563)

Declared In
NSStatusItem.h

image

Returns the image that is displayed at the receiver's position in the status bar.

```
- (NSImage *)image
```

Discussion
Returns nil if an image has not been set.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setImage: (page 2563)
- alternateImage (page 2557)

Declared In
NSStatusItem.h

isEnabled

Returns whether the receiver is enabled and responding to clicks.

```
- (BOOL)isEnabled
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setEnabled:` (page 2563)

Declared In
`NSStatusItem.h`

length

Returns the amount of space allocated to the receiver within its status bar.

```
- (CGFloat)length
```

Discussion
If the status bar is horizontal, the return value is the width of the status item. Besides a physical length, the return value may be `NSSquareStatusItemLength` or `NSVariableStatusItemLength` (see NSStatusBar "Constants" (page 2554)), if the status item size is either determined by the status bar thickness or allowed to vary according to the status item's true size, respectively.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setLength:` (page 2564)
- `statusItemWithLength:` (page 2553) (NSStatusBar)

Declared In
`NSStatusItem.h`

menu

Returns the drop-down menu that is displayed when the receiver is clicked.

```
- (NSMenu *)menu
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMenu:` (page 2564)

Declared In
`NSStatusItem.h`

popUpStatusItemMenu:

Displays a menu under a custom status bar item.

```
- (void)popUpStatusItemMenu:(NSMenu *)menu
```

Parameters

menu

The `NSMenu` object to display.

Discussion

You can use this method to cause a popup menu to appear under a custom status bar item when the user clicks the item. Note that the view of the receiver must exist (that is, it must not be `nil`).

Availability

Available in Mac OS X v10.3 and later.

See Also

- `setMenu:` (page 2564)
- `setView:` (page 2566)

Declared In

`NSStatusItem.h`

sendActionOn:

Sets the conditions on which the receiver sends action messages to its target.

```
- (NSInteger)sendActionOn:(NSInteger)mask
```

Parameters

mask

Takes one or more of the following bit masks described in "Constants" (page 1091) section of the `NSEvent` class reference: `NSLeftMouseUpMask`, `NSLeftMouseDownMask`, `NSLeftMouseDraggedMask`, and `NSPeriodicMask`. Bitwise-OR multiple bit masks.

Return Value

A bit mask containing the previous settings. This bit mask uses the same values as specified in the *mask* parameter.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSStatusItem.h`

setAction:

Sets the selector that is sent to the receiver's target when the receiver is clicked.

```
- (void)setAction:(SEL)action
```

Discussion

If the receiver has a menu set, *action* is not sent to the target when the receiver is clicked; instead, the click causes the menu to appear.

See *Action Messages* for additional information on action messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `action` (page 2557)
- `setMenu:` (page 2564)

Declared In
`NSStatusItem.h`

setAlternateImage:

Sets an alternate image to be displayed when a status bar item is highlighted.

```
- (void)setAlternateImage:(NSImage *)image
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `alternateImage` (page 2557)
- `setImage:` (page 2563)

Declared In
`NSStatusItem.h`

setAttributedTitle:

Sets the attributed string that is displayed at the receiver's position in the status bar.

```
- (void)setAttributedTitle:(NSAttributedString *)title
```

Discussion
If an image is also set, the title appears to the right of the image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `attributedTitle` (page 2558)
- `setImage:` (page 2563)
- `setTitle:` (page 2565)

Declared In
`NSStatusItem.h`

setDoubleAction:

Sets the selector that is sent to the receiver's target when the receiver is double-clicked.

```
- (void)setDoubleAction:(SEL)action
```

Discussion
For the method to have any effect, the receiver's action and target must be set to the class in which the selector is declared. See *Action Messages* for additional information on action messages.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `doubleAction` (page 2558)

Declared In
`NSStatusItem.h`

setEnabled:

Sets whether the receiver is enabled to respond to clicks.

```
- (void)setEnabled:(BOOL)flag
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isEnabled` (page 2559)

Declared In
`NSStatusItem.h`

setHighlightMode:

Sets whether the receiver is highlighted when it is clicked.

```
- (void)setHighlightMode:(BOOL)flag
```

Discussion
The default is `NO`, which means the receiver isn't highlighted when it is clicked.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `highlightMode` (page 2559)

Declared In
`NSStatusItem.h`

setImage:

Sets the image that is displayed at the receiver's position in the status bar to *image*.

```
- (void)setImage:(NSImage *)image
```

Discussion
If a title is also set, the image appears to the left of the title.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `image` (page 2559)
- `setAlternateImage:` (page 2562)
- `setAttributedTitle:` (page 2562)
- `setTitle:` (page 2565)

Declared In
`NSStatusItem.h`

setLength:

Sets the amount of space in the status bar that should be allocated to the receiver.

```
- (void)setLength:(CGFloat)len
```

Parameters
len

If the status bar is horizontal, *len* is the horizontal space to allocate. In addition to a fixed length, *len* can be `NSSquareStatusItemLength` or `NSVariableStatusItemLength` (see "Constants" (page 2554) in the `NSStatusBar` class reference) to allow the status bar to allocate (and adjust) the space according to either the status bar's thickness or the status item's true size.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `length` (page 2560)
- `statusItemWithLength:` (page 2553) (NSStatusBar)

Declared In
`NSStatusItem.h`

setMenu:

Sets the pull-down menu that is displayed when the receiver is clicked.

```
- (void)setMenu:(NSMenu *)menu
```

Parameters
menu

The `NSMenu` object to display.

Discussion
When set, the receiver's single click action behavior is not used. The menu can be removed by setting *menu* to `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `menu` (page 2560)
- `setAction:` (page 2561)
- `setTarget:` (page 2565)
- `popUpStatusItemMenu:` (page 2560)

Declared In
`NSStatusItem.h`

setTarget:

Sets the target object to which the receiver's action message is sent when the receiver is clicked.

```
- (void)setTarget:(id)target
```

Discussion
If the receiver has a menu set, the action is not sent to *target* when the receiver is clicked; instead, the click causes the menu to appear.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `target` (page 2567)
- `setMenu:` (page 2564)

Declared In
`NSStatusItem.h`

setTitle:

Sets the string that is displayed at the receiver's position in the status bar.

```
- (void)setTitle:(NSString *)title
```

Discussion
If an image is also set, the title appears to the right of the image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `title` (page 2567)
- `setAttributedTitle:` (page 2562)
- `setImage:` (page 2563)

Declared In
`NSStatusItem.h`

setToolTip:

Sets the tool tip string that is displayed when the cursor pauses over the receiver.

```
- (void)setToolTip:(NSString *)toolTip
```

Parameters

toolTip

A string that functions as the title of the status item.

Availability
Available in Mac OS X v10.0 and later.

See Also
- toolTip (page 2567)

Declared In
NSStatusItem.h

setView:

Sets the custom view that is displayed at the receiver's position in the status bar.

```
- (void)setView:(NSView *)view
```

Parameters

view

The NSView object representing the custom view.

Discussion
Setting a custom view overrides all the other appearance and behavior settings defined by NSStatusItem. The custom view is responsible for drawing itself and providing its own behaviors, such as processing mouse clicks and sending action messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
- view (page 2567)

Declared In
NSStatusItem.h

statusBar

Returns the status bar in which the receiver is displayed.

```
- (NSStatusBar *)statusBar
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSStatusItem.h

target

Returns the target to which the receiver's action message is sent when the user clicks the receiver.

```
- (id)target
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTarget:` (page 2565)
- `action` (page 2557)

Declared In
`NSStatusItem.h`

title

Returns the string that is displayed at the receiver's position in the status bar.

```
- (NSString *)title
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAttributedTitle:` (page 2562)
- `setTitle:` (page 2565)

Declared In
`NSStatusItem.h`

toolTip

Returns the tool tip string that is displayed when the cursor pauses over the receiver.

```
- (NSString *)toolTip
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setToolTip:` (page 2566)

Declared In
`NSStatusItem.h`

view

Returns the custom view that is displayed at the receiver's position in the status bar.

```
- (NSView *)view
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setView:` (page 2566)

Declared In
`NSStatusItem.h`

NSStepper Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSStepper.h
Companion guide	Steppers
Related sample code	ButtonMadness Quartz 2D Transformer SampleScannerApp

Overview

A stepper consists of two small arrows that can increment and decrement a value that appears beside it, such as a date or time. The illustration below shows a stepper to the right of a text field, which would show the stepper's value.

The `NSStepper` class uses the `NSStepperCell` class to implement its user interface.

Tasks

Specifying Value Range

- `maxValue` (page 2571)
  Returns the maximum value for the receiver.
- `setMaxValue:` (page 2572)
  Sets the maximum value for the receiver

- `minValue` (page 2571)
  Returns the minimum value for the receiver.
- `setMinValue:` (page 2573)
  Sets the minimum value for the receiver
- `increment` (page 2570)
  Returns the amount by which the receiver will change per increment (decrement).
- `setIncrement:` (page 2572)
  Sets the amount by which the receiver will change per increment (decrement).

Specifying How the Stepper Responds

- `autorepeat` (page 2570)
  Returns a Boolean value indicating how the receiver responds to mouse events.
- `setAutorepeat:` (page 2572)
  Sets how the receiver responds to mouse events.
- `valueWraps` (page 2573)
  Returns a Boolean value indicating whether the receiver wraps around the minimum and maximum values.
- `setValueWraps:` (page 2573)
  Sets whether the receiver wraps around the minimum and maximum values.

Instance Methods

autorepeat

Returns a Boolean value indicating how the receiver responds to mouse events.

```
- (BOOL)autorepeat
```

Return Value
`YES` if the first mouse down does one increment (or decrement) and, after a delay of 0.5 seconds, increments (or decrements) at a rate of ten times per second. `NO` if the receiver does one increment (decrement) on a mouse up. The default is `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAutorepeat:` (page 2572)

Declared In
`NSStepper.h`

increment

Returns the amount by which the receiver will change per increment (decrement).

```
- (double)increment
```

Return Value
The amount by which the receiver changes with each increment or decrement. The default is 1.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setIncrement: (page 2572)

Declared In
NSStepper.h

maxValue

Returns the maximum value for the receiver.

```
- (double)maxValue
```

Return Value
The maximum value. The default is 59.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setMaxValue: (page 2572)

Declared In
NSStepper.h

minValue

Returns the minimum value for the receiver.

```
- (double)minValue
```

Return Value
The minimum value. The default is 0.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setMinValue: (page 2573)

Declared In
NSStepper.h

setAutorepeat:

Sets how the receiver responds to mouse events.

```
- (void)setAutorepeat:(BOOL)autorepeat
```

Parameters

autorepeat

If YES, the first mouse down does one increment (decrement) and, after a delay of 0.5 seconds, increments (decrements) at a rate of ten times per second. If *autorepeat* is NO, the receiver does one increment (decrement) on a mouse up.

Availability
Available in Mac OS X v10.0 and later.

See Also
- autorepeat (page 2570)

Declared In
NSStepper.h

setIncrement:

Sets the amount by which the receiver will change per increment (decrement).

```
- (void)setIncrement:(double)increment
```

Parameters

increment

The amount by which the receiver changes with each decrement or increment.

Availability
Available in Mac OS X v10.0 and later.

See Also
- increment (page 2570)

Declared In
NSStepper.h

setMaxValue:

Sets the maximum value for the receiver

```
- (void)setMaxValue:(double)maxValue
```

Parameters

maxValue

The new maximum value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `maxValue` (page 2571)

Declared In
`NSStepper.h`

setMinValue:

Sets the minimum value for the receiver

```
- (void)setMinValue:(double)minValue
```

Parameters

minValue
The new minimum value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `minValue` (page 2571)

Declared In
`NSStepper.h`

setValueWraps:

Sets whether the receiver wraps around the minimum and maximum values.

```
- (void)setValueWraps:(BOOL)valueWraps
```

Parameters

valueWraps
If `YES`, then when incrementing or decrementing, the value wraps around to the minimum or maximum. If *valueWraps* is `NO`, the value stays pinned at the minimum or maximum.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `valueWraps` (page 2573)

Declared In
`NSStepper.h`

valueWraps

Returns a Boolean value indicating whether the receiver wraps around the minimum and maximum values.

```
- (BOOL)valueWraps
```

Return Value
`YES` if, when incrementing or decrementing, the value wraps around to the minimum or maximum. `NO` if the value stays pinned at the minimum or maximum. The default is `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setValueWraps:` (page 2573)

Declared In
`NSStepper.h`

NSStepperCell Class Reference

Inherits from	NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSStepperCell.h
Companion guide	Steppers

Overview

An `NSStepperCell` object controls the appearance and behavior of an `NSStepper` object.

Tasks

Specifying Value Range

- `maxValue` (page 2577)
  Returns the maximum value for the receiver.
- `setMaxValue:` (page 2578)
  Sets the maximum value for the receiver.
- `minValue` (page 2577)
  Returns the minimum value for the receiver.
- `setMinValue:` (page 2578)
  Sets the minimum value for the receiver.
- `increment` (page 2576)
  Returns the amount by which the receiver will change per increment or decrement.
- `setIncrement:` (page 2578)
  Sets the amount by which the receiver will change per increment or decrement.

Specifying How Stepper Cell Responds

- `autorepeat` (page 2576)
  Returns a Boolean value indicating how the receiver responds to mouse events.
- `setAutorepeat:` (page 2577)
  Sets how the receiver responds to mouse events.
- `valueWraps` (page 2579)
  Returns a Boolean value indicating whether the receiver wraps around the minimum and maximum values.
- `setValueWraps:` (page 2579)
  Sets whether the receiver wraps around the minimum and maximum values.

Instance Methods

autorepeat

Returns a Boolean value indicating how the receiver responds to mouse events.

```
- (BOOL)autorepeat
```

Return Value
If `YES`, the first mouse down will do one increment (decrement), and, after a delay of 0.5 seconds, will increment (decrement) at a rate of ten times per second. If `NO`, the receiver will do one increment (decrement) on a mouse up. The default is `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAutorepeat:` (page 2577)

Declared In
`NSStepperCell.h`

increment

Returns the amount by which the receiver will change per increment or decrement.

```
- (double)increment
```

Return Value
The amount by which the receiver changes. The default is 1.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setIncrement:` (page 2578)

Declared In
NSStepperCell.h

maxValue

Returns the maximum value for the receiver.

```
- (double)maxValue
```

Return Value
The maximum value. The default is 59.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setMaxValue: (page 2578)

Declared In
NSStepperCell.h

minValue

Returns the minimum value for the receiver.

```
- (double)minValue
```

Return Value
The minimum value. The default is 0.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setMinValue: (page 2578)

Declared In
NSStepperCell.h

setAutorepeat:

Sets how the receiver responds to mouse events.

```
- (void)setAutorepeat:(BOOL)autorepeat
```

Parameters

autorepeat
If YES, the first mouse down will do one increment (decrement) and, after a delay of 0.5 seconds, will increment (decrement) at a rate of ten times per second. If *autorepeat* is NO, the receiver will do one increment (decrement) on a mouse up.

Availability
Available in Mac OS X v10.0 and later.

See Also
- autorepeat (page 2576)

Declared In
NSStepperCell.h

setIncrement:

Sets the amount by which the receiver will change per increment or decrement.

```
- (void)setIncrement:(double)increment
```

Parameters

increment
The amount by which the receiver changes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- increment (page 2576)

Declared In
NSStepperCell.h

setMaxValue:

Sets the maximum value for the receiver.

```
- (void)setMaxValue:(double)maxValue
```

Parameters

maxValue
The new maximum value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- maxValue (page 2577)

Declared In
NSStepperCell.h

setMinValue:

Sets the minimum value for the receiver.

```
- (void)setMinValue:(double)minValue
```

Parameters

minValue

The new minimum value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- minValue (page 2577)

Declared In
NSStepperCell.h

setValueWraps:

Sets whether the receiver wraps around the minimum and maximum values.

```
- (void)setValueWraps:(BOOL)valueWraps
```

Parameters

valueWraps

If YES, then when incrementing or decrementing, the value will wrap around to the minimum or maximum. If *valueWraps* is NO, the value will stay pinned at the minimum or maximum.

Availability
Available in Mac OS X v10.0 and later.

See Also
- valueWraps (page 2579)

Declared In
NSStepperCell.h

valueWraps

Returns a Boolean value indicating whether the receiver wraps around the minimum and maximum values.

```
- (BOOL)valueWraps
```

Return Value
YES if, when incrementing or decrementing, the value wraps around to the minimum or maximum. NO if the value stays pinned at the minimum or maximum. The default is YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setValueWraps: (page 2579)

Declared In
NSStepperCell.h

NSString Application Kit Additions Reference

Inherits from	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSStringDrawing.h
Companion guides	String Programming Guide Attributed String Programming Guide

Overview

The Application Kit adds three methods to the `NSString` class to support drawing string objects directly in an `NSView` instance: `drawAtPoint:withAttributes:` (page 2582), `drawInRect:withAttributes:` (page 2583), and `sizeWithAttributes:` (page 2585).

The Application Kit adds similar methods to the `NSAttributedString` class, described in *NSAttributedString Application Kit Additions Reference*. The two drawing methods draw a string object with a single set of attributes that apply to the entire string. To draw a string with multiple attributes, such as multiple text fonts, you must use an attributed string.

Tasks

Drawing String Objects

`- drawAtPoint:withAttributes:` (page 2582)
Draws the receiver with the font and other display characteristics of the given attributes, at the specified point in the currently focused view.

`- drawInRect:withAttributes:` (page 2583)
Draws the receiver with the font and other display characteristics of the given attributes, within the specified rectangle in the currently focused NSView.

`- drawWithRect:options:attributes:` (page 2584)
Draws the receiver with the specified options and other display characteristics of the given attributes, within the specified rectangle in the current graphics context.

`- sizeWithAttributes:` (page 2585)
Returns the bounding box size the receiver occupies when drawn with the given attributes.

Getting the Bounding Rect of Rendered Strings

Instance Methods

boundingRectWithSize:options:attributes:

Calculates and returns the bounding rect for the receiver drawn using the given options and display characteristics, within the specified rectangle in the current graphics context.

```
- (NSRect)boundingRectWithSize:(NSSize)size options:(NSStringDrawingOptions)options
    attributes:(NSDictionary *)attributes
```

Parameters

size
The size of the rectangle to draw in.

options
String drawing options.

attributes
A dictionary of text attributes to be applied to the string. These are the same attributes that can be applied to an `NSAttributedString` object, but in the case of `NSString` objects, the attributes apply to the entire string, rather than ranges within the string.

Return Value
The bounding rect for the receiver drawn using the given options and display characteristics. The rect origin returned from this method is the first glyph origin.

Availability
Available in Mac OS X v10.4 and later.

See Also
`drawInRect:withAttributes:` (page 2583)

Declared In
`NSStringDrawing.h`

drawAtPoint:withAttributes:

Draws the receiver with the font and other display characteristics of the given attributes, at the specified point in the currently focused view.

```
- (void)drawAtPoint:(NSPoint)aPoint withAttributes:(NSDictionary *)attributes
```

Parameters

aPoint

The origin for the bounding box for drawing the string. If the focused view is flipped, the origin is the upper-left corner of the drawing bounding box; otherwise, the origin is the lower-left corner.

attributes

A dictionary of text attributes to be applied to the string. These are the same attributes that can be applied to an `NSAttributedString` object, but in the case of `NSString` objects, the attributes apply to the entire string, rather than ranges within the string.

Discussion

The width (height for vertical layout) of the rendering area is unlimited, unlike `drawInRect:withAttributes:` (page 2583), which uses a bounding rectangle. As a result, this method renders the text in a single line.

You should only invoke this method when an `NSView` object has focus.

Availability

Available in Mac OS X v10.0 and later.

See Also

`drawInRect:withAttributes:` (page 2583) (NSView)

`drawWithRect:options:attributes:` (page 2584)

`lockFocus` (page 3187)

Related Sample Code

CIVideoDemoGL

ClockControl

CocoaVideoFrameToGWorld

FunHouse

SpecialPictureProtocol

Declared In

`NSStringDrawing.h`

drawInRect:withAttributes:

Draws the receiver with the font and other display characteristics of the given attributes, within the specified rectangle in the currently focused NSView.

```
- (void)drawInRect:(NSRect)aRect withAttributes:(NSDictionary *)attributes
```

Parameters

aRect

The rectangle in which to draw the string.

attributes

A dictionary of text attributes to be applied to the string. These are the same attributes that can be applied to an `NSAttributedString` object, but in the case of `NSString` objects, the attributes apply to the entire string, rather than ranges within the string.

Discussion

The rendering area is bounded by *aRect*, unlike `drawAtPoint:withAttributes:` (page 2582), which has an unlimited width. As a result, this method renders the text in multiple lines.

You should only invoke this method when an NSView has focus.

Availability
Available in Mac OS X v10.0 and later.

See Also
`drawAtPoint:withAttributes:` (page 2582) (NSView)
`drawWithRect:options:attributes:` (page 2584)
`lockFocus` (page 3187)

Related Sample Code
FilterDemo
SampleRaster
SurfaceVertexProgram
VertexPerformanceDemo
Worm

Declared In
`NSStringDrawing.h`

drawWithRect:options:attributes:

Draws the receiver with the specified options and other display characteristics of the given attributes, within the specified rectangle in the current graphics context.

```
- (void)drawWithRect:(NSRect)rect options:(NSStringDrawingOptions)options
    attributes:(NSDictionary *)attributes
```

Parameters

rect
The rectangle in which to draw the string.

options
String drawing options.

attributes
A dictionary of text attributes to be applied to the string. These are the same attributes that can be applied to an `NSAttributedString` object, but in the case of `NSString` objects, the attributes apply to the entire string, rather than ranges within the string.

Discussion
The *rect* argument's `origin` field specifies the rendering origin. The point is interpreted as the baseline origin by default. With `NSStringDrawingUsesLineFragmentOrigin`, it is interpreted as the upper left corner of the line fragment rect. The `size` field specifies the text container size. The `width` part of the size field specifies the maximum line fragment width if larger than `0.0`. The `height` defines the maximum size that can be occupied with text if larger than `0.0` and `NSStringDrawingUsesLineFragmentOrigin` is specified. If `NSStringDrawingUsesLineFragmentOrigin` is not specified, height is ignored and considered to be single-line rendering (`NSLineBreakByWordWrapping` and `NSLineBreakByCharWrapping` are treated as `NSLineBreakByClipping`).

You should only invoke this method when there is a current graphics context.

Availability
Available in Mac OS X v10.4 and later.

See Also
`drawAtPoint:withAttributes:` (page 2582) (NSView)
`drawInRect:withAttributes:` (page 2583)
`lockFocus` (page 3187)

Declared In
`NSStringDrawing.h`

sizeWithAttributes:

Returns the bounding box size the receiver occupies when drawn with the given attributes.

```
- (NSSize)sizeWithAttributes:(NSDictionary *)attributes
```

Parameters
attributes
A dictionary of text attributes to be applied to the string. These are the same attributes that can be applied to an `NSAttributedString` object, but in the case of `NSString` objects, the attributes apply to the entire string, rather than ranges within the string.

Return Value
The bounding box size the receiver occupies when drawn with *attributes*.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CIVideoDemoGL
CocoaVideoFrameToGWorld
PhotoSearch
SampleRaster
SpecialPictureProtocol

Declared In
`NSStringDrawing.h`

Constants

NSStringDrawingOptions

The following constants are provided as rendering options for a string when it is drawn.

```
enum {
    NSStringDrawingUsesLineFragmentOrigin = (1 << 0),
    NSStringDrawingUsesFontLeading = (1 << 1),
    NSStringDrawingDisableScreenFontSubstitution = (1 << 2),
    NSStringDrawingUsesDeviceMetrics = (1 << 3),
    NSStringDrawingOneShot = (1 << 4),
    NSStringDrawingTruncatesLastVisibleLine = (1 << 5)
};
typedef NSInteger NSStringDrawingOptions;
```

Constants

`NSStringDrawingUsesLineFragmentOrigin`

The specified origin is the line fragment origin, not the baseline origin.

Available in Mac OS X v10.4 and later.

Declared in `NSStringDrawing.h`.

`NSStringDrawingUsesFontLeading`

Uses the font leading for calculating line heights.

Available in Mac OS X v10.4 and later.

Declared in `NSStringDrawing.h`.

`NSStringDrawingDisableScreenFontSubstitution`

Disable screen font substitution (equivalent to `[NSLayoutManager setUsesScreenFonts:NO]`).

Available in Mac OS X v10.4 and later.

Declared in `NSStringDrawing.h`.

`NSStringDrawingUsesDeviceMetrics`

Uses image glyph bounds instead of typographic bounds.

Available in Mac OS X v10.4 and later.

Declared in `NSStringDrawing.h`.

`NSStringDrawingOneShot`

Suppresses caching layout information.

Available in Mac OS X v10.4 and later.

Declared in `NSStringDrawing.h`.

`NSStringDrawingTruncatesLastVisibleLine`

Truncates and adds the ellipsis character to the last visible line if the text doesn't fit into the bounds specified.

This option is ignored if `NSStringDrawingUsesLineFragmentOrigin` is not also set. In addition, the line break mode must be either `NSLineBreakByWordWrapping` or `NSLineBreakByCharWrapping` for this option to take effect. The line break mode can be specified in a paragraph style passed in the attributes dictionary argument of the drawing methods.

Available in Mac OS X v10.5 and later.

Declared in `NSStringDrawing.h`.

NSTableColumn Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTableColumn.h
Companion guide	Table View Programming Guide
Related sample code	AnimatedTableView MixMash MyPhoto QTKitMovieShuffler STUCAuthoringDeviceCocoaSample

Overview

An NSTableColumn stores the display characteristics and attribute identifier for a column in an NSTableView. The NSTableColumn determines the width and width limits, resizability, and editability of its column in the NSTableView. It also stores two NSCell objects: the header cell, which is used to draw the column header, and the data cell, used to draw the values for each row. You can control the display of the column by setting the subclasses of NSCell used and by setting the font and other display characteristics for these NSCells. For example, you can use the default NSTextFieldCell for displaying string values or substitute an NSImageCell to display pictures.

Adopted Protocols

NSCoding

- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Creating an NSTableColumn

- `initWithIdentifier:` (page 2592)
  Initializes a newly created NSTableColumn with *identifier* as its identifier and with an NSTextFieldCell as its data cell.

Setting the NSTableView

- `setTableView:` (page 2599)
  Sets *aTableView* as the receiver's NSTableView.
- `tableView` (page 2601)
  Returns the NSTableView the receiver belongs to.

Controlling Size

- `setWidth:` (page 2600)
  Sets the receiver's width to *newWidth*.
- `width` (page 2601)
  Returns the width of the receiver.
- `setMinWidth:` (page 2598)
  Sets the receiver's minimum width to *minWidth*, also adjusting the current width if it's less than this value.
- `minWidth` (page 2594)
  Returns the minimum width for the receiver.
- `setMaxWidth:` (page 2597)
  Sets the receiver's maximum width to *maxWidth*, also adjusting the current width if it's greater than this value.
- `maxWidth` (page 2593)
  Returns the maximum width of the receiver.
- `setResizingMask:` (page 2598)
  Sets the resizing mask for the receiver to *resizingMask*.
- `resizingMask` (page 2594)
  Returns the receiver's resizing mask.
- `sizeToFit` (page 2600)
  Resizes the receiver to fit the width of its header cell.

Setting Component Cells

- `setHeaderCell:` (page 2596)
  Sets the NSCell used to draw the receiver's header to *aCell*.

- `headerCell` (page 2591)
  Returns the NSTableHeaderCell object used to draw the header of the receiver.
- `setDataCell:` (page 2595)
  Sets the NSCell used by the NSTableView to draw individual values for the receiver to *aCell*.
- `dataCell` (page 2590)
  Returns the NSCell object used by the NSTableView to draw values for the receiver.
- `dataCellForRow:` (page 2590)
  Returns the NSCell object used by the NSTableView to draw values for the receiver.

Setting the Identifier

- `setIdentifier:` (page 2597)
  Sets the receiver's identifier to *anObject*.
- `identifier` (page 2591)
  Returns the object used by the data source to identify the attribute corresponding to the receiver.

Controlling Editability

- `setEditable:` (page 2595)
  Controls whether the user can edit cells in the receiver by double-clicking them.
- `isEditable` (page 2592)
  Returns `YES` if the user can edit cells associated with the receiver by double-clicking the column in the NSTableView, `NO` otherwise.

Sorting

- `setSortDescriptorPrototype:` (page 2599)
  Sets the receiver's sort descriptor prototype.
- `sortDescriptorPrototype` (page 2600)
  Returns the receiver's sort descriptor prototype.

Setting Column Visibility

- `isHidden` (page 2593)
  Returns a Boolean value that indicates whether the receiver is hidden.
- `setHidden:` (page 2596)
  Sets whether the receiver is hidden.

Setting Tool Tips

- `setHeaderToolTip:` (page 2596)
  Sets the tooltip string that is displayed when the cursor pauses over the header cell of the receiver.

- `headerToolTip` (page 2591)
  Returns the tooltip string that is displayed when the cursor pauses over the header cell of the receiver.

Deprecated Methods

- `isResizable` (page 2593) Deprecated in Mac OS X v10.4
  Returns `YES` if the user is allowed to resize the receiver in its NSTableView, `NO` otherwise.
- `setResizable:` (page 2598) Deprecated in Mac OS X v10.4
  Sets whether the user can resize the receiver in its NSTableView.

Instance Methods

dataCell

Returns the NSCell object used by the NSTableView to draw values for the receiver.

```
- (id)dataCell
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDataCell:` (page 2595)

Related Sample Code
AnimatedTableView
DragNDropOutlineView
FunHouse
SourceView

Declared In
`NSTableColumn.h`

dataCellForRow:

Returns the NSCell object used by the NSTableView to draw values for the receiver.

```
- (id)dataCellForRow:(NSInteger)row
```

Discussion
NSTableView always calls this method. By default, this method just calls `dataCell` (page 2590). Subclassers can override if they need to potentially use different cells for different rows. Subclasses should expect this method to be invoked with *row* equal to –1 in cases where no actual row is involved but the table view needs to get some generic cell info.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTableColumn.h

headerCell

Returns the NSTableHeaderCell object used to draw the header of the receiver.

```
- (id)headerCell
```

Discussion
You can set the column title by sending `setStringValue:` (page 596) to this object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setHeaderCell:` (page 2596)

Related Sample Code
CapabilitiesSample

Declared In
NSTableColumn.h

headerToolTip

Returns the tooltip string that is displayed when the cursor pauses over the header cell of the receiver.

```
- (NSString *)headerToolTip
```

Return Value
The tooltip displayed when the cursor pauses over the header cell of the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setHeaderToolTip:` (page 2596)

Declared In
NSTableColumn.h

identifier

Returns the object used by the data source to identify the attribute corresponding to the receiver.

```
- (id)identifier
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setIdentifier:` (page 2597)

Related Sample Code
DragNDropOutlineView
MyPhoto
NewsReader
NSOperationSample
QTAudioContextInsert

Declared In
`NSTableColumn.h`

initWithIdentifier:

Initializes a newly created NSTableColumn with *identifier* as its identifier and with an NSTextFieldCell as its data cell.

```
- (id)initWithIdentifier:(id)identifier
```

Discussion
Send `setStringValue:` (page 596) to the header cell to set the column title. This method is the designated initializer for the NSTableColumn class. Returns an initialized object.

See the NSTableView class specification for information on identifiers.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setIdentifier:` (page 2597)

Declared In
`NSTableColumn.h`

isEditable

Returns `YES` if the user can edit cells associated with the receiver by double-clicking the column in the NSTableView, `NO` otherwise.

```
- (BOOL)isEditable
```

Discussion
You can initiate editing programmatically regardless of this setting with NSTableView's `editColumn:row:withEvent:select:` (page 2633) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setEditable:` (page 2595)

Declared In
NSTableColumn.h

isHidden

Returns a Boolean value that indicates whether the receiver is hidden.

```
- (BOOL)isHidden
```

Return Value
YES if the receiver is hidden, otherwise NO.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setHidden: (page 2596)

Declared In
NSTableColumn.h

isResizable

Returns YES if the user is allowed to resize the receiver in its NSTableView, NO otherwise. (Deprecated in Mac OS X v10.4.)

```
- (BOOL)isResizable
```

Discussion
You can change the size programmatically regardless of this setting.

This method is deprecated. You should use resizingMask (page 2594) instead.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- setWidth: (page 2600)
- setMinWidth: (page 2598)
- setMaxWidth: (page 2597)
- setResizable: (page 2598)

Declared In
NSTableColumn.h

maxWidth

Returns the maximum width of the receiver.

```
- (CGFloat)maxWidth
```

Discussion
The receiver's width can't be made larger than this size either by the user or programmatically.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `minWidth` (page 2594)
- `width` (page 2601)
- `setMaxWidth:` (page 2597)
- `autoresizesAllColumnsToFit` (page 2622) (NSTableView)

Declared In
`NSTableColumn.h`

minWidth

Returns the minimum width for the receiver.

```
- (CGFloat)minWidth
```

Discussion
The receiver's width can't be made less than this size either by the user or programmatically.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `maxWidth` (page 2593)
- `width` (page 2601)
- `setMinWidth:` (page 2598)
- `autoresizesAllColumnsToFit` (page 2622) (NSTableView)

Declared In
`NSTableColumn.h`

resizingMask

Returns the receiver's resizing mask.

```
- (NSUInteger)resizingMask
```

Discussion
See "`Resizing Modes`" (page 2602) for a description of the resizing mask constants.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setResizingMask:` (page 2598)

Declared In
NSTableColumn.h

setDataCell:

Sets the NSCell used by the NSTableView to draw individual values for the receiver to *aCell*.

```
- (void)setDataCell:(NSCell *)aCell
```

Discussion
You can use this method to control the font, alignment, and other text attributes for an NSTableColumn. You can also assign a cell to display things other than text—for example, an NSImageCell to display images.

Availability
Available in Mac OS X v10.0 and later.

See Also
- dataCell (page 2590)

Related Sample Code
EnhancedDataBurn
ImageBackground
QTKitMovieShuffler
SourceView

Declared In
NSTableColumn.h

setEditable:

Controls whether the user can edit cells in the receiver by double-clicking them.

```
- (void)setEditable:(BOOL)flag
```

Discussion
If *flag* is YES a double click initiates editing; if *flag* is NO it merely sends the double-click action to the NSTableView's target. You can initiate editing programmatically regardless of this setting with NSTableView's editColumn:row:withEvent:select: (page 2633) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isEditable (page 2592)

Related Sample Code
SimpleCocoaMovie
SimpleCocoaMovieQT

Declared In
NSTableColumn.h

setHeaderCell:

Sets the NSCell used to draw the receiver's header to *aCell*.

```
- (void)setHeaderCell:(NSCell *)aCell
```

Discussion
aCell should never be `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `headerCell` (page 2591)

Declared In
`NSTableColumn.h`

setHeaderToolTip:

Sets the tooltip string that is displayed when the cursor pauses over the header cell of the receiver.

```
- (void)setHeaderToolTip:(NSString *)string
```

Parameters
string
A string that functions as the tooltip for the header cell of the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `headerToolTip` (page 2591)

Declared In
`NSTableColumn.h`

setHidden:

Sets whether the receiver is hidden.

```
- (void)setHidden:(BOOL)hidden
```

Parameters
hidden
YES if the receiver is to be hidden, otherwise NO.

Discussion
Columns which are hidden still exist in the tableview's `tableColumns` (page 2670) array and are included in the tableview's `numberOfColumns` (page 2641) count.

The hidden state of the receiver is stored when the tableview autosaves the NSTableColumn state.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `isHidden` (page 2593)

Declared In
`NSTableColumn.h`

setIdentifier:

Sets the receiver's identifier to *anObject*.

```
- (void)setIdentifier:(id)anObject
```

Discussion
This object is used by the data source to identify the attribute corresponding to the NSTableColumn.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `identifier` (page 2591)

Related Sample Code
CapabilitiesSample

Declared In
`NSTableColumn.h`

setMaxWidth:

Sets the receiver's maximum width to *maxWidth*, also adjusting the current width if it's greater than this value.

```
- (void)setMaxWidth:(CGFloat)maxWidth
```

Discussion
The NSTableView can be made no wider than this size, either by the user or programmatically.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMinWidth:` (page 2598)
- `setWidth:` (page 2600)
- `maxWidth` (page 2593)
- `autoresizesAllColumnsToFit` (page 2622) (NSTableView)

Declared In
`NSTableColumn.h`

setMinWidth:

Sets the receiver's minimum width to *minWidth*, also adjusting the current width if it's less than this value.

```
- (void)setMinWidth:(CGFloat)minWidth
```

Discussion
The NSTableView can be made no less wide than this size, either by the user or programmatically.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMaxWidth:` (page 2597)
- `setWidth:` (page 2600)
- `minWidth` (page 2594)
- `autoresizesAllColumnsToFit` (page 2622) (NSTableView)

Declared In
`NSTableColumn.h`

setResizable:

Sets whether the user can resize the receiver in its NSTableView. (Deprecated in Mac OS X v10.4.)

```
- (void)setResizable:(BOOL)flag
```

Discussion
If *flag* is `YES` the user can resize the receiver; if *flag* is `NO` the user can't resize it. You can always set the size programmatically.

This method is deprecated. You should use `setResizingMask:` (page 2598) instead.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- `isResizable` (page 2593)
- `setWidth:` (page 2600)
- `setMinWidth:` (page 2598)
- `setMaxWidth:` (page 2597)

Declared In
`NSTableColumn.h`

setResizingMask:

Sets the resizing mask for the receiver to *resizingMask*.

```
- (void)setResizingMask:(NSUInteger)resizingMask
```

Discussion
If *resizingMask* is 0, the column is not resizable. See "Resizing Modes" (page 2602) for the appropriate mask values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- resizingMask (page 2594)

Declared In
NSTableColumn.h

setSortDescriptorPrototype:

Sets the receiver's sort descriptor prototype.

```
- (void)setSortDescriptorPrototype:(NSSortDescriptor *)sortDescriptor
```

Discussion
A table column is considered sortable if it has a sort descriptor that specifies the sorting direction, a key to sort by, and a selector defining how to sort.

Availability
Available in Mac OS X v10.3 and later.

See Also
- sortDescriptorPrototype (page 2600)

Declared In
NSTableColumn.h

setTableView:

Sets *aTableView* as the receiver's NSTableView.

```
- (void)setTableView:(NSTableView *)aTableView
```

Discussion
You should never need to invoke this method; it's invoked automatically when you add an NSTableColumn to an NSTableView.

Availability
Available in Mac OS X v10.0 and later.

See Also
- tableView (page 2601)
- addTableColumn: (page 2619) (NSTableView)

Declared In
NSTableColumn.h

setWidth:

Sets the receiver's width to *newWidth*.

```
- (void)setWidth:(CGFloat)newWidth
```

Discussion
If *newWidth* exceeds the minimum or maximum width, it's adjusted to the appropriate limiting value. Marks the NSTableView as needing display.

This method posts `NSTableViewColumnDidResizeNotification` (page 2679) on behalf of the receiver's NSTableView.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `width` (page 2601)
- `setMinWidth:` (page 2598)
- `setMaxWidth:` (page 2597)
- `autoresizesAllColumnsToFit` (page 2622) (NSTableView)

Declared In
`NSTableColumn.h`

sizeToFit

Resizes the receiver to fit the width of its header cell.

```
- (void)sizeToFit
```

Discussion
If the maximum width is less than the width of the header, the maximum is increased to the header's width. Similarly, if the minimum width is greater than the width of the header, the minimum is reduced to the header's width. Marks the NSTableView as needing display if the width actually changes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `width` (page 2601)
- `minWidth` (page 2594)
- `maxWidth` (page 2593)
- `autoresizesAllColumnsToFit` (page 2622) (NSTableView)

Declared In
`NSTableColumn.h`

sortDescriptorPrototype

Returns the receiver's sort descriptor prototype.

```
- (NSSortDescriptor *)sortDescriptorPrototype
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- setSortDescriptorPrototype: (page 2599)

Declared In
NSTableColumn.h

tableView

Returns the NSTableView the receiver belongs to.

```
- (NSTableView *)tableView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTableView: (page 2599)

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel

Declared In
NSTableColumn.h

width

Returns the width of the receiver.

```
- (CGFloat)width
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
FunHouse

Declared In
NSTableColumn.h

Constants

Resizing Modes

These constants specify the resizing modes available for the table column. You specify either NSTableColumnNoResizing or a resizing mask created using the C bitwise OR operator. These values are then passed as the parameter to the `setResizingMask:` (page 2598) method.

```
enum {     NSTableColumnNoResizing = 0,      NSTableColumnAutoresizingMask = ( 1
<< 0 ),     NSTableColumnUserResizingMask = ( 1 << 1 ), };
```

Constants

`NSTableColumnNoResizing`

Prevents the table column from resizing.

Available in Mac OS X v10.4 and later.

Declared in `NSTableColumn.h`.

`NSTableColumnAutoresizingMask`

Allows the table column to resize automatically in response to resizing the tableview. Enabling this option is the same as enabling the "Live Resizable" option in Interface Builder. The resizing behavior for the table view is set using the `NSTableView` method `setColumnAutoresizingStyle:` (page 2659).

Available in Mac OS X v10.4 and later.

Declared in `NSTableColumn.h`.

`NSTableColumnUserResizingMask`

Allows the table column to be resized explicitly by the user. Enabling this option is the same as enabling the "User Resizable" option in Interface Builder.

Available in Mac OS X v10.4 and later.

Declared in `NSTableColumn.h`.

Declared In

`NSTableColumn.h`

NSTableHeaderCell Class Reference

Inherits from	NSTextFieldCell : NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTableHeaderCell.h
Companion guide	Table View Programming Guide

Overview

An NSTableHeaderCell is used by an NSTableHeaderView to draw its column headers. See the NSTableView class specification for more information on how it's used.

Subclasses of NSTableHeaderCell can override `drawInteriorWithFrame:inView:` (page 553), `editWithFrame:inView:editor:delegate:event:` (page 555), and `highlight:withFrame:inView:` (page 560) to change the way headers appear. See the NSCell class specification, and the following description, for information on these methods.

Tasks

Sorting

- `drawSortIndicatorWithFrame:inView:ascending:priority:` (page 2604)
  Draws a sorting indicator given a *cellFrame* contained inside *controlView*.
- `sortIndicatorRectForBounds:` (page 2604)
  Returns the location to display the sorting indicator given *theRect*.

Instance Methods

drawSortIndicatorWithFrame:inView:ascending:priority:

Draws a sorting indicator given a *cellFrame* contained inside *controlView*.

```
- (void)drawSortIndicatorWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
    ascending:(BOOL)ascending priority:(NSInteger)priority
```

Discussion
If *priority* is 0, this is the primary sort indicator. If *ascending* is YES, a "^" indicator will be drawn. Override this method to customize the sorting user interface.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTableHeaderCell.h

sortIndicatorRectForBounds:

Returns the location to display the sorting indicator given *theRect*.

```
- (NSRect)sortIndicatorRectForBounds:(NSRect)theRect
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTableHeaderCell.h

NSTableHeaderView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTableHeaderView.h
Companion guide	Table View Programming Guide

Overview

An NSTableHeaderView is used by an NSTableView to draw headers over its columns and to handle mouse events in those headers.

NSTableHeaderView uses NSTableHeaderCell to implement its user interface.

Tasks

Setting the Table View

- `setTableView:` (page 2608)
  Sets *aTableView* as the receiver's NSTableView.
- `tableView` (page 2608)
  Returns the NSTableView the receiver belongs to.

Checking Altered Columns

- `draggedColumn` (page 2606)
  If the user is dragging a column in the receiver, returns the index of that column.
- `draggedDistance` (page 2607)
  If the user is dragging a column in the receiver, returns the column's horizontal distance from its original position.

- `resizedColumn` (page 2607)
  If the user is resizing a column in the receiver, returns the index of that column.

Utility Methods

- `columnAtPoint:` (page 2606)
  Returns the index of the column whose header lies under *aPoint* in the receiver, or –1 if no such column is found.
- `headerRectOfColumn:` (page 2607)
  Returns the rectangle containing the header tile for the column at *columnIndex*.

Instance Methods

columnAtPoint:

Returns the index of the column whose header lies under *aPoint* in the receiver, or –1 if no such column is found.

```
- (NSInteger)columnAtPoint:(NSPoint)aPoint
```

Discussion
aPoint is expressed in the receiver's coordinate system.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableHeaderView.h`

draggedColumn

If the user is dragging a column in the receiver, returns the index of that column.

```
- (NSInteger)draggedColumn
```

Discussion
Otherwise returns –1.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `draggedDistance` (page 2607)

Declared In
`NSTableHeaderView.h`

draggedDistance

If the user is dragging a column in the receiver, returns the column's horizontal distance from its original position.

```
- (CGFloat)draggedDistance
```

Discussion
Otherwise the return value is meaningless.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `draggedColumn` (page 2606)

Declared In
`NSTableHeaderView.h`

headerRectOfColumn:

Returns the rectangle containing the header tile for the column at *columnIndex*.

```
- (NSRect)headerRectOfColumn:(NSInteger)columnIndex
```

Discussion
Raises an `NSInternalInconsistencyException` if *columnIndex* is out of bounds.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rectOfColumn:` (page 2644) (NSTableView)

Declared In
`NSTableHeaderView.h`

resizedColumn

If the user is resizing a column in the receiver, returns the index of that column.

```
- (NSInteger)resizedColumn
```

Discussion
Otherwise returns –1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableHeaderView.h`

setTableView:

Sets *aTableView* as the receiver's NSTableView.

```
- (void)setTableView:(NSTableView *)aTableView
```

Discussion
You should never need to invoke this method; it's invoked automatically when you set the header view for an NSTableView.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setHeaderView: (page 2664) (NSTableView)

Declared In
NSTableHeaderView.h

tableView

Returns the NSTableView the receiver belongs to.

```
- (NSTableView *)tableView
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTableHeaderView.h

NSTableView Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations NSTextViewDelegate NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTableView.h
Companion guides	Table View Programming Guide Drag and Drop Programming Topics for Cocoa
Related sample code	CoreRecipes DemoMonkey MyPhoto QTAudioContextInsert With and Without Bindings

Class at a Glance

An `NSTableView` object displays record-oriented data in a table and allows the user to edit values and resize and rearrange columns.

Principal Attributes

- A data source
- Table columns

Commonly Used Methods

`dataSource` (page 2627)
Returns the object providing the data that the table view displays.

`tableColumns` (page 2670)
Returns the `NSTableColumn` objects representing attributes for the table view.

`selectedColumn` (page 2650)
Returns the index of the selected column.

`selectedRow` (page 2651)
Returns the index of the selected row.

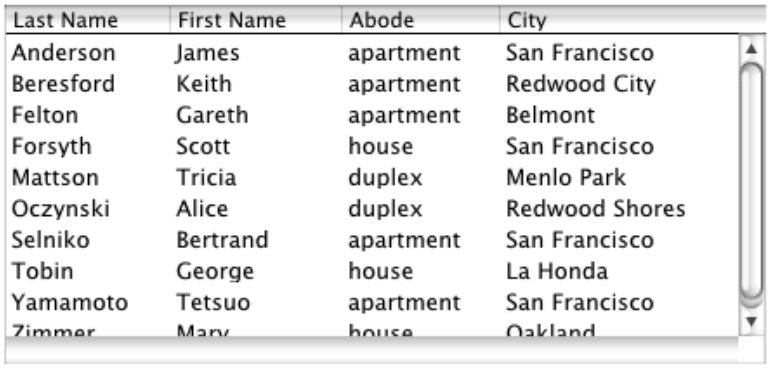
`numberOfRows` (page 2642)
Returns the number of rows in the table view.

`reloadData` (page 2645)
Informs the table view that data has changed and needs to be retrieved and displayed again.

Overview

An `NSTableView` object displays data for a set of related records, with rows representing individual records and columns representing the attributes of those records.

A table view is usually displayed in a scroll view, like this:



Last Name	First Name	Abode	City
Anderson	James	apartment	San Francisco
Beresford	Keith	apartment	Redwood City
Felton	Gareth	apartment	Belmont
Forsyth	Scott	house	San Francisco
Mattson	Tricia	duplex	Menlo Park
Oczynski	Alice	duplex	Redwood Shores
Selniko	Bertrand	apartment	San Francisco
Tobin	George	house	La Honda
Yamamoto	Tetsuo	apartment	San Francisco
Zimmer	Mary	house	Oakland

A table view does not store its own data, instead it retrieves data values as needed from a data source to which it has a weak reference (see Communicating With Objects). You should not, therefore, try to directly set data values programmatically in the table view; instead you should modify the values in the data source and allow the changes to be reflected in the table view. See the `NSTableDataSource protocol`, which declares the methods that an `NSTableView` object uses to access the contents of its data source object.

Tasks

Setting the Data Source

`- setDataSource:` (page 2660)
Sets the receiver's data source to a given object.

- `dataSource` (page 2627)
  Returns the object that provides the data displayed by the receiver.

Loading Data

- `reloadData` (page 2645)
  Marks the receiver as needing redisplay, so it will reload the data for visible cells and draw the new values.
- `reloadDataForRowIndexes:columnIndexes:` (page 2646)
  Reloads the data for only the specified rows and columns.

Target-action Behavior

- `setDoubleAction:` (page 2661)
  Sets the message sent to the target when the user double-clicks an uneditable cell or a column header to a given selector.
- `doubleAction` (page 2630)
  Returns the message sent to the target when the user double-clicks a column header or an uneditable cell.
- `clickedColumn` (page 2624)
  Returns the index of the column the user clicked to trigger an action message.
- `clickedRow` (page 2624)
  Returns the index of the row the user clicked to trigger an action message.

Configuring Behavior

- `setAllowsColumnReordering:` (page 2654)
  Controls whether the user can drag column headers to reorder columns.
- `allowsColumnReordering` (page 2619)
  Returns a Boolean value that indicates whether the receiver allows the user to rearrange columns by dragging their headers.
- `setAllowsColumnResizing:` (page 2655)
  Controls whether the user can resize columns by dragging between headers.
- `allowsColumnResizing` (page 2619)
  Returns a Boolean value that indicates whether the receiver allows the user to resize columns by dragging between their headers.
- `setAllowsMultipleSelection:` (page 2656)
  Controls whether the user can select more than one row or column at a time.
- `allowsMultipleSelection` (page 2621)
  Returns a Boolean value that indicates whether the receiver allows the user to select more than one column or row at a time.
- `setAllowsEmptySelection:` (page 2656)
  Controls whether the receiver allows zero rows or columns to be selected.

- `allowsEmptySelection` (page 2620)
  Returns a Boolean value that indicates whether the receiver allows the user to select zero columns or rows.
- `setAllowsColumnSelection:` (page 2655)
  Controls whether the user can select an entire column by clicking its header.
- `allowsColumnSelection` (page 2620)
  Returns a Boolean value that indicates whether the receiver allows the user to select columns by clicking their headers.

Setting Display Attributes

- `setIntercellSpacing:` (page 2666)
  Sets the width and height between cells to those in a given `NSSize` struct.
- `intercellSpacing` (page 2638)
  Returns the horizontal and vertical spacing between cells.
- `setRowHeight:` (page 2666)
  Sets the height for rows to a given value.
- `rowHeight` (page 2647)
  Returns the height of each row in the receiver.
- `setBackgroundColor:` (page 2658)
  Sets the receiver's background color to a given color.
- `backgroundColor` (page 2623)
  Returns the color used to draw the background of the receiver.
- `setUsesAlternatingRowBackgroundColors:` (page 2667)
  Sets whether the receiver uses the standard alternating row colors for its background.
- `usesAlternatingRowBackgroundColors` (page 2673)
  Returns a Boolean value that indicates whether the receiver uses the standard alternating row colors for its background.
- `selectionHighlightStyle` (page 2653)
  Returns the selection highlight style used by the receiver to indicate row and column selection.
- `setSelectionHighlightStyle:` (page 2667)
  Sets the selection highlight style used by the receiver to indicate row and column selection.
- `setGridColor:` (page 2664)
  Sets the color used to draw grid lines to a given color.
- `gridColor` (page 2636)
  Returns the color used to draw grid lines.
- `setGridStyleMask:` (page 2664)
  Sets the grid style mask to specify if no grid lines, vertical grid lines, or horizontal grid lines should be displayed.
- `gridStyleMask` (page 2636)
  Returns the receiver's grid style mask.
- `indicatorImageInTableColumn:` (page 2638)
  Returns the indicator image of a given table column.

- `setIndicatorImage:inTableColumn:` (page 2665)
  Sets the indicator image of *aTableColumn* to *anImage*.

Column Management

- `addTableColumn:` (page 2619)
  Adds a given column as the last column of the receiver.
- `removeTableColumn:` (page 2646)
  Removes a given column from the receiver.
- `moveColumn:toColumn:` (page 2640)
  Moves the column and heading at a given index to a new given index.
- `tableColumns` (page 2670)
  Returns an array containing the the `NSTableColumn` objects in the receiver.
- `columnWithIdentifier:` (page 2627)
  Returns the index of the first column in the receiver whose identifier is equal to a given identifier.
- `tableColumnWithIdentifier:` (page 2670)
  Returns the `NSTableColumn` object for the first column whose identifier is equal to a given object.

Selecting Columns and Rows

- `selectColumnIndexes:byExtendingSelection:` (page 2650)
  Sets the column selection using *indexes*.
- `selectedColumn` (page 2650)
  Returns the index of the last column selected or added to the selection.
- `selectedColumnIndexes` (page 2651)
  Returns an index set containing the indexes of the selected columns.
- `deselectColumn:` (page 2629)
  Deselects the column at a given index if it's selected.
- `numberOfSelectedColumns` (page 2642)
  Returns the number of selected columns.
- `isColumnSelected:` (page 2639)
  Returns a Boolean value that indicates whether the column at a given index is selected.
- `selectRowIndexes:byExtendingSelection:` (page 2654)
  Sets the row selection using *indexes*.
- `selectedRow` (page 2651)
  Returns the index of the last row selected or added to the selection.
- `selectedRowIndexes` (page 2652)
  Returns an index set containing the indexes of the selected rows.
- `deselectRow:` (page 2630)
  Deselects the row at a given index if it's selected.
- `numberOfSelectedRows` (page 2642)
  Returns the number of selected rows.

- `isRowSelected:` (page 2639)
  Returns a Boolean value that indicates whether the row at a given index is selected.
- `selectAll:` (page 2649)
  Selects all rows or all columns, according to whether rows or columns were most recently selected.
- `deselectAll:` (page 2628)
  Deselects all selected rows or columns if empty selection is allowed; otherwise does nothing.

Managing Type Select

- `allowsTypeSelect` (page 2621)
  Returns a Boolean value that indicates whether the receiver allows the user to type characters to select rows.
- `setAllowsTypeSelect:` (page 2657)
  Sets whether the receiver allows the user to type characters to select rows.

Getting and Setting Column Focus

- `focusedColumn` (page 2635)
  Returns the currently focused column.
- `setFocusedColumn:` (page 2663)
  Sets the currently focused column to the specified index.
- `shouldFocusCell:atColumn:row:` (page 2668)
  Returns whether the fully cell at the specified row and column can be made the focused cell or not.

Table Dimensions

- `numberOfColumns` (page 2641)
  Returns the number of columns in the receiver.
- `numberOfRows` (page 2642)
  Returns the number of rows in the receiver.

Displaying Cell

- `preparedCellAtColumn:row:` (page 2643)
  Returns the fully prepared cell that the receiver will use for drawing or processing of the specified row and column.

Editing Cells

- `editColumn:row:withEvent:select:` (page 2633)
  Edits the cell at *`columnIndex`* and *`rowIndex`*, selecting its entire contents if *`flag`* is `YES`.
- `editedColumn` (page 2634)
  Returns the index of the column being edited.

- `editedRow` (page 2634)
  Returns the index of the row being edited.
- `performClickOnCellAtColumn:row:` (page 2643)
  Performs a click action on the cell at the specified row and column.

Setting Auxiliary Views

- `setHeaderView:` (page 2664)
  Sets the receiver's header view to a given header view.
- `headerView` (page 2637)
  Returns the `NSTableHeaderView` object used to draw headers over columns.
- `setCornerView:` (page 2659)
  Sets the receiver's corner view to a given view.
- `cornerView` (page 2627)
  Returns the view used to draw the area to the right of the column headers and above the vertical scroller of the enclosing scroll view.

Layout Support

- `rectOfColumn:` (page 2644)
  Returns the rectangle containing the column at a given index.
- `rectOfRow:` (page 2645)
  Returns the rectangle containing the row at a given index.
- `rowsInRect:` (page 2648)
  Returns a range of indices for the rows that lie wholly or partially within the vertical boundaries of a given rectangle.
- `columnIndexesInRect:` (page 2626)
  Returns the indexes of the receiver's columns that intersect the specified rectangle.
- `columnAtPoint:` (page 2625)
  Returns the index of the column a given point lies in.
- `rowAtPoint:` (page 2647)
  Returns the index of the row a given point lies in.
- `frameOfCellAtColumn:row:` (page 2635)
  Returns a rectangle locating the cell that lies at the intersection of *`columnIndex`* and *`rowIndex`*.
- `columnAutoresizingStyle` (page 2625)
  Returns the receiver's column autoresizing style.
- `setColumnAutoresizingStyle:` (page 2659)
  Sets the column autoresizing style of the receiver to a given style.
- `sizeLastColumnToFit` (page 2669)
  Resizes the last column if there's room so the receiver fits exactly within its enclosing clip view.
- `noteNumberOfRowsChanged` (page 2641)
  Informs the receiver that the number of records in its data source has changed.
- `tile` (page 2673)
  Properly sizes the receiver and its header view and marks it as needing display.

- `sizeToFit` (page 2669)
  Changes the width of columns in the receiver so all columns are visible.
- `noteHeightOfRowsWithIndexesChanged:` (page 2640)
  Informs the receiver that the rows specified in *indexSet* have changed height.
- `columnsInRect:` (page 2626) Deprecated in Mac OS X v10.5
  Returns a range of indices for the receiver's columns that lie wholly or partially within the horizontal boundaries of a given rectangle. (Deprecated. Use `columnIndexesInRect:` (page 2626) instead.)

Drawing

- `drawRow:clipRect:` (page 2633)
  Draws the cells for the row at *rowIndex* in the columns that intersect *clipRect*.
- `drawGridInClipRect:` (page 2632)
  Draws the grid lines within *aRect*, using the grid color set with `setGridColor:` (page 2664).
- `highlightSelectionInClipRect:` (page 2638)
  Highlights the region of the receiver in *clipRect*.
- `drawBackgroundInClipRect:` (page 2632)
  Draws the background in the clip rect specified by *clipRect*.

Scrolling

- `scrollRowToVisible:` (page 2648)
  Scrolls the receiver vertically in an enclosing NSClipView so the row specified by *rowIndex* is visible.
- `scrollColumnToVisible:` (page 2648)
  Scrolls the receiver and header view horizontally in an enclosing NSClipView so the column specified by *columnIndex* is visible.

Persistence

- `autosaveName` (page 2622)
  Returns the name under which table information is automatically saved.
- `autosaveTableColumns` (page 2623)
  Returns a Boolean value that indicates whether the order and width of the receiver's columns are automatically saved.
- `setAutosaveName:` (page 2657)
  Sets the name under which table information is automatically saved to *name*.
- `setAutosaveTableColumns:` (page 2658)
  Sets whether the order and width of this table view's columns are automatically saved.

Setting the Delegate

- `setDelegate:` (page 2660)
  Sets the receiver's delegate to a given object.

- `delegate` (page 2628)
  Returns the receiver's delegate.

Highlightable Column Headers

- `highlightedTableColumn` (page 2637)
  Returns the table column highlighted in the receiver.
- `setHighlightedTableColumn:` (page 2665)
  Sets *`aTableColumn`* to be the currently highlighted column header.

Dragging

- `dragImageForRowsWithIndexes:tableColumns:event:offset:` (page 2631)
  Computes and returns an image to use for dragging.
- `canDragRowsWithIndexes:atPoint:` (page 2623)
  Returns whether the receiver allows dragging the rows at *`rowIndexes`* with a drag initiated at *`mousedDownPoint`*.
- `setDraggingSourceOperationMask:forLocal:` (page 2662)
  Sets the default operation mask returned by `draggingSourceOperationMaskForLocal:` to *`mask`*.
- `setVerticalMotionCanBeginDrag:` (page 2668)
  Sets whether vertical motion is treated as a drag or selection change to *`flag`*.
- `verticalMotionCanBeginDrag` (page 2674)
  Returns whether vertical motion is treated as a drag or selection change.
- `draggingDestinationFeedbackStyle` (page 2631)
  Returns the feedback style displayed when the user drags over the table view.
- `setDraggingDestinationFeedbackStyle:` (page 2662)
  Sets the feedback style displayed when the user drags over the table view.
- `setDropRow:dropOperation:` (page 2663)
  Used if you wish to "retarget" the proposed drop.

Sorting

- `setSortDescriptors:` (page 2667)
  Sets the receiver's sort descriptors to the `NSSortDescriptor` objects in *`array`*.
- `sortDescriptors` (page 2670)
  Returns the receiver's sort descriptors.

Text Delegate Methods

- `textShouldBeginEditing:` (page 2672)
  Queries the delegate using `control:textShouldBeginEditing:`, returning the delegate's response, or simply returning `YES` to allow editing of *`textObject`* if the delegate doesn't respond to that method.

- `textDidBeginEditing:` (page 2671)
  Posts an `NSControlTextDidBeginEditingNotification` (page 847) to the default notification center.
- `textDidChange:` (page 2671)
  Sends `textDidChange:` (page 2671) to the edited cell and posts an `NSControlTextDidChangeNotification` (page 848) to the default notification center.
- `textShouldEndEditing:` (page 2673)
  Validates the *`textObject`* cell being edited and queries the delegate using `control:textShouldEndEditing:`, returning the delegate's response if it responds to that method.
- `textDidEndEditing:` (page 2672)
  Updates the data source based on the newly edited value and selects another cell for editing if possible according to the character that ended editing (Return, Tab, Backtab).

Deprecated Methods

- `tableView:writeRows:toPasteboard:` (page 2674) *delegate method*
  Writes the specified rows to the specified pasteboard. (Deprecated. This method has been deprecated. You should implement `tableView:writeRowsWithIndexes:toPasteboard:` (page 3825) instead.)
- `drawsGrid` (page 2633) Deprecated in Mac OS X v10.3
  Returns a Boolean value that indicates whether the receiver draws a grid. (Deprecated. Use `gridStyleMask` (page 2636) instead.)
- `selectColumn:byExtendingSelection:` (page 2649) Deprecated in Mac OS X v10.3
  Selects a column at a given index, optionally extending any existing selection. (Deprecated. Use `selectColumnIndexes:byExtendingSelection:` (page 2650) instead.)
- `selectedColumnEnumerator` (page 2651) Deprecated in Mac OS X v10.3
  This method has been deprecated. (Deprecated. Use `selectedColumnIndexes` (page 2651) instead.)
- `selectedRowEnumerator` (page 2652) Deprecated in Mac OS X v10.3
  This method has been deprecated. (Deprecated. Use `selectedRowIndexes` (page 2652) instead.)
- `selectRow:byExtendingSelection:` (page 2653) Deprecated in Mac OS X v10.3
  Selects a row at a given index, optionally extending any existing selection. (Deprecated. Use `selectRowIndexes:byExtendingSelection:` (page 2654) instead.)
- `setDrawsGrid:` (page 2662) Deprecated in Mac OS X v10.3
  Sets whether the receiver draws a grid. (Deprecated. Use `setGridStyleMask:` (page 2664) instead.)
- `autoresizesAllColumnsToFit` (page 2622) Deprecated in Mac OS X v10.4
  Returns `YES` if the receiver proportionally resizes its columns to fit when its superview's frame changes, `NO` if it only resizes the last column. (Deprecated. Use `columnAutoresizingStyle` (page 2625) instead.)
- `dragImageForRows:event:dragImageOffset:` (page 2631) Deprecated in Mac OS X v10.4
  Computes and returns an image to use for dragging. (Deprecated. Use `dragImageForRowsWithIndexes:tableColumns:event:offset:` (page 2631) instead.)
- `setAutoresizesAllColumnsToFit:` (page 2657) Deprecated in Mac OS X v10.4
  Controls whether the receiver proportionally resizes its columns to fit when its superview's frame changes. (Deprecated. Use `setColumnAutoresizingStyle:` (page 2659) instead.)

Instance Methods

addTableColumn:

Adds a given column as the last column of the receiver.

```
- (void)addTableColumn:(NSTableColumn *)aColumn
```

Parameters

aColumn

The column to add to the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sizeLastColumnToFit` (page 2669)
- `removeTableColumn:` (page 2646)

Declared In
`NSTableView.h`

allowsColumnReordering

Returns a Boolean value that indicates whether the receiver allows the user to rearrange columns by dragging their headers.

```
- (BOOL)allowsColumnReordering
```

Return Value
`YES` to allow the user to rearrange columns by dragging their headers, otherwise `NO`.

Discussion
The default is `YES`. You can rearrange columns programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `moveColumn:toColumn:` (page 2640)
- `setAllowsColumnReordering:` (page 2654)

Declared In
`NSTableView.h`

allowsColumnResizing

Returns a Boolean value that indicates whether the receiver allows the user to resize columns by dragging between their headers.

```
- (BOOL)allowsColumnResizing
```

Return Value
YES if the receiver allows the user to resize columns by dragging between their headers, otherwise NO.

Discussion
The default is YES. You can resize columns programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setWidth: (page 2600) (NSTableColumn)
- setAllowsColumnResizing: (page 2655)

Declared In
NSTableView.h

allowsColumnSelection

Returns a Boolean value that indicates whether the receiver allows the user to select columns by clicking their headers.

```
- (BOOL)allowsColumnSelection
```

Return Value
YES if the receiver allows the user to select columns by clicking their headers, otherwise NO.

Discussion
The default is NO. You can select columns programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectColumn:byExtendingSelection: (page 2649)
- allowsColumnReordering (page 2619)
- setAllowsColumnSelection: (page 2655)

Declared In
NSTableView.h

allowsEmptySelection

Returns a Boolean value that indicates whether the receiver allows the user to select zero columns or rows.

```
- (BOOL)allowsEmptySelection
```

Return Value
YES if the receiver allows the user to select zero columns or rows, otherwise NO.

Discussion
The default is YES.

You cannot set an empty selection programmatically if this setting is NO, unlike with the other settings that affect selection behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
- deselectAll: (page 2628)
- deselectColumn: (page 2629)
- deselectRow: (page 2630)
- setAllowsEmptySelection: (page 2656)

Declared In
NSTableView.h

allowsMultipleSelection

Returns a Boolean value that indicates whether the receiver allows the user to select more than one column or row at a time.

```
- (BOOL)allowsMultipleSelection
```

Return Value
YES if the receiver allows the user to select more than one column or row at a time, otherwise NO.

Discussion
The default is NO. You can select multiple columns or rows programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectColumn:byExtendingSelection: (page 2649)
- selectRow:byExtendingSelection: (page 2653)
- setAllowsMultipleSelection: (page 2656)

Declared In
NSTableView.h

allowsTypeSelect

Returns a Boolean value that indicates whether the receiver allows the user to type characters to select rows.

```
- (BOOL)allowsTypeSelect
```

Return Value
YES if the receiver allows type selection, otherwise NO.

Discussion
The default value is YES.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setAllowsTypeSelect:` (page 2657)

Declared In
`NSTableView.h`

autoresizesAllColumnsToFit

Returns `YES` if the receiver proportionally resizes its columns to fit when its superview's frame changes, `NO` if it only resizes the last column. (Deprecated in Mac OS X v10.4. Use `columnAutoresizingStyle` (page 2625) instead.)

```
- (BOOL)autoresizesAllColumnsToFit
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- `columnAutoresizingStyle` (page 2625)
- `setColumnAutoresizingStyle:` (page 2659)

Declared In
`NSTableView.h`

autosaveName

Returns the name under which table information is automatically saved.

```
- (NSString *)autosaveName
```

Return Value
The name under which table information is automatically saved. If no name has been set, returns `nil`.

Discussion
The table information is saved separately for each user and for each application that user uses.

Note that even when a table view has an autosave name, it may not be saving table information automatically. To check whether table information is being saved automatically, use `autosaveTableColumns` (page 2623).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `autosaveTableColumns` (page 2623)
- `setAutosaveName:` (page 2657)

Declared In
`NSTableView.h`

autosaveTableColumns

Returns a Boolean value that indicates whether the order and width of the receiver's columns are automatically saved.

```
- (BOOL)autosaveTableColumns
```

Discussion
The table information is saved separately for each user and for each application that user uses. Note that if `autosaveName` (page 2622) returns `nil`, this setting is ignored and table information isn't saved.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `autosaveName` (page 2622)
- `setAutosaveTableColumns:` (page 2658)
- `setAutosaveName:` (page 2657)

Declared In
`NSTableView.h`

backgroundColor

Returns the color used to draw the background of the receiver.

```
- (NSColor *)backgroundColor
```

Return Value
The color used to draw the background of the receiver.

Discussion
The default background color is light gray.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBackgroundColor:` (page 2658)

Declared In
`NSTableView.h`

canDragRowsWithIndexes:atPoint:

Returns whether the receiver allows dragging the rows at *`rowIndexes`* with a drag initiated at *`mousedDownPoint`*.

```
- (BOOL)canDragRowsWithIndexes:(NSIndexSet *)rowIndexes
    atPoint:(NSPoint)mouseDownPoint
```

Discussion
Return `NO` to disallow the drag.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTableView.h`

clickedColumn

Returns the index of the column the user clicked to trigger an action message.

```
- (NSInteger)clickedColumn
```

Return Value
The index of the column the user clicked to trigger an action message. Returns `-1` if the user clicked in an area of the table view not occupied by columns.

Discussion
The return value of this method is meaningful only in the target's implementation of the action or double-action method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `clickedRow` (page 2624)
- `setAction:` (page 828) (`NSControl`)
- `setDoubleAction:` (page 2661)

Related Sample Code
AnimatedTableView
DragNDropOutlineView
PhotoSearch

Declared In
`NSTableView.h`

clickedRow

Returns the index of the row the user clicked to trigger an action message.

```
- (NSInteger)clickedRow
```

Return Value
The index of the row the user clicked to trigger an action message. Returns `-1` if the user clicked in an area of the table view not occupied by table rows.

Discussion
The return value of this method is meaningful only in the target's implementation of the action or double-action method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `clickedColumn` (page 2624)
- `setAction:` (page 828) (`NSControl`)
- `setDoubleAction:` (page 2661)

Related Sample Code
AnimatedTableView
DragNDropOutlineView
PhotoSearch
WhackedTV

Declared In
`NSTableView.h`

columnAtPoint:

Returns the index of the column a given point lies in.

```
- (NSInteger)columnAtPoint:(NSPoint)aPoint
```

Parameters
aPoint
A point in the coordinate system of the receiver.

Return Value
The index of the column *aPoint* lies in, or `-1` if *aPoint* lies outside the receiver's bounds.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rowAtPoint:` (page 2647)

Declared In
`NSTableView.h`

columnAutoresizingStyle

Returns the receiver's column autoresizing style.

```
- (NSTableViewColumnAutoresizingStyle)columnAutoresizingStyle
```

Return Value
The receiver's column autoresizing style. For possible values, see "Autoresizing Styles" (page 2677).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setColumnAutoresizingStyle:` (page 2659)

Declared In
NSTableView.h

columnIndexesInRect:

Returns the indexes of the receiver's columns that intersect the specified rectangle.

```
- (NSIndexSet *)columnIndexesInRect:(NSRect)rect
```

Parameters

rect

The rectangle in the receiver's coordinate system to test for column enclosure.

Return Value
New NSIndexSet object containing the indexes of the receiver's columns that intersect with *rect*.

Discussion
Columns that return YES for the NSTableColumn method isHidden (page 2593) are excluded from the results.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTableView.h

columnsInRect:

Returns a range of indices for the receiver's columns that lie wholly or partially within the horizontal boundaries of a given rectangle. (Deprecated in Mac OS X v10.5. Use columnIndexesInRect: (page 2626) instead.)

```
- (NSRange)columnsInRect:(NSRect)aRect
```

Parameters

aRect

A rectangle in the coordinate system of the receiver.

Return Value
A range of indices for the receiver's columns that lie wholly or partially within the horizontal boundaries of *aRect*. If the width or height of *aRect* is 0, returns an NSRange whose length is 0.

Discussion
The location of the range is the first such column's index, and the length is the number of columns that lie in *aRect*.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

See Also
- rowsInRect: (page 2648)

Declared In
NSTableView.h

columnWithIdentifier:

Returns the index of the first column in the receiver whose identifier is equal to a given identifier.

```
- (NSInteger)columnWithIdentifier:(id)anObject
```

Parameters

anObject

A column identifier.

Return Value

The index of the first column in the receiver whose identifier is equal to *anObject* (when compared using `isEqual:`) or `-1` if no columns are found with the specified identifier.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- tableColumnWithIdentifier:` (page 2670)

Declared In

`NSTableView.h`

cornerView

Returns the view used to draw the area to the right of the column headers and above the vertical scroller of the enclosing scroll view.

```
- (NSView *)cornerView
```

Return Value

The view used to draw the area to the right of the column headers and above the vertical scroller of the enclosing `NSScrollView` object.

Discussion

This is by default a simple view that merely fills in its frame, but you can replace it with a custom view using `setCornerView:` (page 2659).

Availability

Available in Mac OS X v10.0 and later.

See Also

`- headerView` (page 2637)

Declared In

`NSTableView.h`

dataSource

Returns the object that provides the data displayed by the receiver.

```
- (id < NSTableViewDataSource >)dataSource
```

Return Value
The object that provides the data displayed by the receiver.

Discussion
See Using a Table Data Source and the `NSTableDataSource` informal `protocol` specification for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setDataSource:` (page 2660)

Related Sample Code
CocoaAUHost

Declared In
`NSTableView.h`

delegate

Returns the receiver's delegate.

```
- (id < NSTableViewDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setDelegate:` (page 2660)

Related Sample Code
AnimatedTableView
People

Declared In
`NSTableView.h`

deselectAll:

Deselects all selected rows or columns if empty selection is allowed; otherwise does nothing.

```
- (void)deselectAll:(id)sender
```

Parameters
sender
Typically the object that sent the message.

Discussion
Posts `NSTableViewSelectionDidChangeNotification` (page 2679) to the default notification center if the selection does in fact change.

As a target-action method, `deselectAll:` checks with the delegate before changing the selection, using `selectionShouldChangeInTableView:` (page 3829).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsEmptySelection` (page 2620)
- `selectAll:` (page 2649)
- `selectColumn:byExtendingSelection:` (page 2649)
- `selectionShouldChangeInTableView:` (page 3829) (NSTableViewDelegate)

Related Sample Code
DragNDropOutlineView
EnhancedAudioBurn
EnhancedDataBurn
QTKitMovieShuffler

Declared In
`NSTableView.h`

deselectColumn:

Deselects the column at a given index if it's selected.

```
- (void)deselectColumn:(NSInteger)columnIndex
```

Parameters
columnIndex
The index of the column to deselect.

Discussion
Deselects the column at *columnIndex* if it's selected, regardless of whether empty selection is allowed.

If the selection does in fact change, posts `NSTableViewSelectionDidChangeNotification` (page 2679) to the default notification center.

If the indicated column was the last column selected by the user, the column nearest it effectively becomes the last selected column. In case of a tie, priority is given to the column on the left.

This method doesn't check with the delegate before changing the selection.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedColumn` (page 2650)
- `allowsEmptySelection` (page 2620)
- `selectRow:byExtendingSelection:` (page 2653)

Declared In
NSTableView.h

deselectRow:

Deselects the row at a given index if it's selected.

```
- (void)deselectRow:(NSInteger)rowIndex
```

Parameters

rowIndex
The index of the row to deselect.

Discussion
Deselects the row at *rowIndex* if it's selected, regardless of whether empty selection is allowed.

If the selection does in fact change, posts `NSTableViewSelectionDidChangeNotification` (page 2679) to the default notification center.

If the indicated row was the last row selected by the user, the row nearest it effectively becomes the last selected row. In case of a tie, priority is given to the row above.

This method doesn't check with the delegate before changing the selection.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedRow` (page 2651)
- `allowsEmptySelection` (page 2620)

Related Sample Code
XMLBrowser

Declared In
NSTableView.h

doubleAction

Returns the message sent to the target when the user double-clicks a column header or an uneditable cell.

```
- (SEL)doubleAction
```

Return Value
The message the receiver sends to its target when the user double-clicks a column header or an uneditable cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `action` (page 814) (`NSControl`)
- `target` (page 844) (`NSControl`)

- setDoubleAction: (page 2661)

Declared In
NSTableView.h

draggingDestinationFeedbackStyle

Returns the feedback style displayed when the user drags over the table view.

```
- (NSTableViewDraggingDestinationFeedbackStyle)draggingDestinationFeedbackStyle
```

Return Value
The dragging feedback style. See "NSTableViewDraggingDestinationFeedbackStyle" (page 2675) for the possible values.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setDraggingDestinationFeedbackStyle: (page 2662)

Declared In
NSTableView.h

dragImageForRows:event:dragImageOffset:

Computes and returns an image to use for dragging. (Deprecated in Mac OS X v10.4. Use dragImageForRowsWithIndexes:tableColumns:event:offset: (page 2631) instead.)

```
- (NSImage *)dragImageForRows:(NSArray *)dragRows event:(NSEvent *)dragEvent
    dragImageOffset:(NSPointPointer)dragImageOffset
```

Discussion
Override this to return a custom image. *dragRows* represents the rows participating in the drag. *dragEvent* is a reference to the mouse-down event that began the drag. *dragImageOffset* is an in/out parameter.

This method is called with *dragImageOffset* set to NSZeroPoint, but it can be modified to reposition the returned image. A *dragImageOffset* of NSZeroPoint will cause the image to be centered under the cursor.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
NSTableView.h

dragImageForRowsWithIndexes:tableColumns:event:offset:

Computes and returns an image to use for dragging.

```
- (NSImage *)dragImageForRowsWithIndexes:(NSIndexSet *)dragRows tableColumns:(NSArray
    *)tableColumns event:(NSEvent *)dragEvent offset:(NSPointPointer)dragImageOffset
```

Discussion
Override this to return a custom image. *dragRows* represents the rows participating in the drag. *tableColumns* represents the table columns that should be in the output image. *dragEvent* is a reference to the mouse-down event that began the drag. *dragImageOffset* is an in/out parameter.

This method is called with *dragImageOffset* set to NSZeroPoint, but it can be modified to reposition the returned image. A *dragImageOffset* of NSZeroPoint will cause the image to be centered under the cursor.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTableView.h

drawBackgroundInClipRect:

Draws the background in the clip rect specified by *clipRect*.

```
- (void)drawBackgroundInClipRect:(NSRect)clipRect
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTableView.h

drawGridInClipRect:

Draws the grid lines within *aRect*, using the grid color set with setGridColor: (page 2664).

```
- (void)drawGridInClipRect:(NSRect)aRect
```

Discussion
This method draws a grid regardless of whether the receiver is set to draw one automatically.

Subclasses can override this method to draw grid lines other than the standard ones.

Availability
Available in Mac OS X v10.0 and later.

See Also
- gridColor (page 2636)
- setIntercellSpacing: (page 2666)
- drawsGrid (page 2633)
- drawRow:clipRect: (page 2633)
- highlightSelectionInClipRect: (page 2638)

Declared In
NSTableView.h

drawRow:clipRect:

Draws the cells for the row at *rowIndex* in the columns that intersect *clipRect*.

```
- (void)drawRow:(NSInteger)rowIndex clipRect:(NSRect)clipRect
```

Discussion
Sends `tableView:willDisplayCell:forTableColumn:row:` (page 3840) to the delegate before drawing each cell.

Subclasses can override this method to customize their appearance.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `columnsInRect:` (page 2626)
- `highlightSelectionInClipRect:` (page 2638)
- `drawGridInClipRect:` (page 2632)

Declared In
`NSTableView.h`

drawsGrid

Returns a Boolean value that indicates whether the receiver draws a grid. (Deprecated in Mac OS X v10.3. Use `gridStyleMask` (page 2636) instead.)

```
- (BOOL)drawsGrid
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

Declared In
`NSTableView.h`

editColumn:row:withEvent:select:

Edits the cell at *columnIndex* and *rowIndex*, selecting its entire contents if *flag* is `YES`.

```
- (void)editColumn:(NSInteger)columnIndex row:(NSInteger)rowIndex withEvent:(NSEvent
    *)theEvent select:(BOOL)flag
```

Discussion
This method is invoked automatically in response to user actions; you should rarely need to invoke it directly. *theEvent* is usually the mouse event that triggered editing; it can be `nil` when starting an edit programmatically.

This method scrolls the receiver so that the cell is visible, sets up the field editor, and sends `selectWithFrame:inView:editor:delegate:start:length:` (page 575) and `editWithFrame:inView:editor:delegate:event:` (page 555) to the field editor's NSCell object with the NSTableView as the text delegate.

The row at *rowIndex* must be selected prior to calling `editColumn:row:withEvent:select:`, or an exception will be raised.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `editedColumn` (page 2634)
- `editedRow` (page 2634)

Related Sample Code
DragNDropOutlineView
IdentitySample
SourceView

Declared In
`NSTableView.h`

editedColumn

Returns the index of the column being edited.

```
- (NSInteger)editedColumn
```

Return Value
If sent during `editColumn:row:withEvent:select:` (page 2633), the index of the column being edited; otherwise `-1`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AnimatedTableView
People

Declared In
`NSTableView.h`

editedRow

Returns the index of the row being edited.

```
- (NSInteger)editedRow
```

Return Value
If sent during `editColumn:row:withEvent:select:` (page 2633), the index of the row being edited; otherwise `-1`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AnimatedTableView
People

Declared In
NSTableView.h

focusedColumn

Returns the currently focused column.

```
- (NSInteger)focusedColumn
```

Return Value
The index of the column, or -1 if there is no focused column

Discussion
The focus interaction will always be on the `selectedRow` (page 2651) of the table. If the `selectedRow` (page 2651) is a full width cell, then `focusedColumn` will return `1` when focused..

Availability
Available in Mac OS X v10.6 and later.

See Also
- `shouldFocusCell:atColumn:row:` (page 2668)
- `setFocusedColumn:` (page 2663)

Declared In
NSTableView.h

frameOfCellAtColumn:row:

Returns a rectangle locating the cell that lies at the intersection of *columnIndex* and *rowIndex*.

```
- (NSRect)frameOfCellAtColumn:(NSInteger)columnIndex row:(NSInteger)rowIndex
```

Parameters

columnIndex
The index of the column containing the cell whose rectangle you want.

rowIndex
The index of the row containing the cell whose rectangle you want.

Return Value
A rectangle locating the cell that lies at the intersection of *columnIndex* and *rowIndex*. Returns `NSZeroRect` if *columnIndex* or *rowIndex* is greater than the number of columns or rows in the receiver.

Discussion
You can use this method to update a single cell more efficiently than sending the table view a reloadData (page 2645) message.

```
[aTableView setNeedsDisplayInRect:[aTableView frameOfCellAtColumn:column
row:row]];
```

The result of this method is used in a `drawWithFrame:inView:` (page 554) message to the table column's data cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rectOfColumn:` (page 2644)
- `rectOfRow:` (page 2645)

Related Sample Code
AnimatedTableView
DragNDropOutlineView
PhotoSearch

Declared In
`NSTableView.h`

gridColor

Returns the color used to draw grid lines.

```
- (NSColor *)gridColor
```

Return Value
The color used to draw grid lines.

Discussion
The default color is gray.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawsGrid` (page 2633)
- `drawGridInClipRect:` (page 2632)
- `setGridColor:` (page 2664)

Declared In
`NSTableView.h`

gridStyleMask

Returns the receiver's grid style mask.

```
- (NSUInteger)gridStyleMask
```

Return Value
The receiver's grid style mask. Possible return values are described in "Grid styles" (page 2676).

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setGridStyleMask:` (page 2664)

Declared In
`NSTableView.h`

headerView

Returns the `NSTableHeaderView` object used to draw headers over columns.

```
- (NSTableHeaderView *)headerView
```

Return Value
The `NSTableHeaderView` object used to draw headers over columns, or `nil` if the receiver has no header view

Discussion
See The Parts of a Table and the `NSTableHeaderView` class specification for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHeaderView:` (page 2664)

Declared In
`NSTableView.h`

highlightedTableColumn

Returns the table column highlighted in the receiver.

```
- (NSTableColumn *)highlightedTableColumn
```

Return Value
The table column highlighted in the receiver.

Discussion
A highlightable column header can be used in conjunction with row selection to highlight a particular column of the table. An example of this is how the Mail application indicates the currently sorted column.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHighlightedTableColumn:` (page 2665)

Declared In
`NSTableView.h`

highlightSelectionInClipRect:

Highlights the region of the receiver in *clipRect*.

```
- (void)highlightSelectionInClipRect:(NSRect)clipRect
```

Discussion
This method is invoked before `drawRow:clipRect:` (page 2633).

Subclasses can override this method to change the manner in which they highlight selections.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- drawGridInClipRect:` (page 2632)

Declared In
`NSTableView.h`

indicatorImageInTableColumn:

Returns the indicator image of a given table column.

```
- (NSImage *)indicatorImageInTableColumn:(NSTableColumn *)aTableColumn
```

Parameters

aTableColumn
A table column in the receiver.

Discussion
An indicator image is an arbitrary (small) image that is rendered on the right side of the column header. An example of its use is in Mail to indicate the sorting direction of the currently sorted column in a mailbox.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setIndicatorImage:inTableColumn:` (page 2665)

Declared In
`NSTableView.h`

intercellSpacing

Returns the horizontal and vertical spacing between cells.

```
- (NSSize)intercellSpacing
```

Return Value
The horizontal and vertical spacing between cells.

Discussion
The default spacing is `(3.0, 2.0)`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDrawsGrid:` (page 2662)
- `setIntercellSpacing:` (page 2666)

Related Sample Code
MP3 Player

Declared In
`NSTableView.h`

isColumnSelected:

Returns a Boolean value that indicates whether the column at a given index is selected.

```
- (BOOL)isColumnSelected:(NSInteger)columnIndex
```

Parameters

columnIndex
The index of the column to test.

Return Value
`YES` if the column at *columnIndex* is selected, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedColumn` (page 2650)
- `selectedColumnEnumerator` (page 2651)
- `selectColumn:byExtendingSelection:` (page 2649)

Declared In
`NSTableView.h`

isRowSelected:

Returns a Boolean value that indicates whether the row at a given index is selected.

```
- (BOOL)isRowSelected:(NSInteger)rowIndex
```

Parameters

rowIndex
The index of the row to test.

Return Value
`YES` if the row at *rowIndex* is selected, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedRow` (page 2651)
- `selectedRowEnumerator` (page 2652)
- `selectRow:byExtendingSelection:` (page 2653)

Related Sample Code
DragNDropOutlineView

Declared In
`NSTableView.h`

moveColumn:toColumn:

Moves the column and heading at a given index to a new given index.

```
- (void)moveColumn:(NSInteger)columnIndex toColumn:(NSInteger)newIndex
```

Parameters

columnIndex
The current index of the column to move.

newIndex
The new index for the moved column.

Discussion
This method posts `NSTableViewColumnDidMoveNotification` (page 2678) to the default notification center.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

noteHeightOfRowsWithIndexesChanged:

Informs the receiver that the rows specified in *indexSet* have changed height.

```
- (void)noteHeightOfRowsWithIndexesChanged:(NSIndexSet *)indexSet
```

Discussion
If the delegate implements `tableView:heightOfRow:` (page 3831) this method immediately re-tiles the table view using the row heights the delegate provides.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTableView.h`

noteNumberOfRowsChanged

Informs the receiver that the number of records in its data source has changed.

```
- (void)noteNumberOfRowsChanged
```

Discussion
This method allows the receiver to update the scrollers in its scroll view without actually reloading data into the receiver. It's useful for a data source that continually receives data in the background over a period of time, in which case the table view can remain responsive to the user while the data is received.

See the `NSTableDataSource` informal protocol specification for information on the messages an `NSTableView` object sends to its data source.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `reloadData` (page 2645)
- `numberOfRowsInTableView:` (`NSTableDataSource` informal protocol)

Related Sample Code
NewsReader

Declared In
`NSTableView.h`

numberOfColumns

Returns the number of columns in the receiver.

```
- (NSInteger)numberOfColumns
```

Return Value
The number of columns in the receiver.

Discussion
The value returned includes table columns that are currently hidden.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfRows` (page 2642)

Related Sample Code
NSOperationSample

Declared In
`NSTableView.h`

numberOfRows

Returns the number of rows in the receiver.

```
- (NSInteger)numberOfRows
```

Return Value
The number of rows in the receiver.

Discussion
Typically you should not ask the table view how many rows it has; instead you should interrogate the table view's data source.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfColumns` (page 2641)
- `numberOfRowsInTableView:` (`NSTableDataSource` informal protocol)

Related Sample Code
EnhancedAudioBurn
iSpend
MP3 Player
PDFKitLinker2
QTAudioExtractionPanel

Declared In
`NSTableView.h`

numberOfSelectedColumns

Returns the number of selected columns.

```
- (NSInteger)numberOfSelectedColumns
```

Return Value
The number of selected columns.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfSelectedRows` (page 2642)
- `selectedColumnEnumerator` (page 2651)

Declared In
`NSTableView.h`

numberOfSelectedRows

Returns the number of selected rows.

```
- (NSInteger)numberOfSelectedRows
```

Return Value
The number of selected rows.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfSelectedColumns` (page 2642)
- `selectedRowEnumerator` (page 2652)

Related Sample Code
DragNDropOutlineView
EnhancedAudioBurn

Declared In
`NSTableView.h`

performClickOnCellAtColumn:row:

Performs a click action on the cell at the specified row and column.

```
- (void)performClickOnCellAtColumn:(NSInteger)column row:(NSInteger)row
```

Parameters

column
The column of the cell.

row
The row of the cell.

Discussion
Acquires the `preparedCellAtColumn:row:` (page 2643), copies it, invokes `performClick:` (page 573) or `performClickWithFrame:inView:` (page 2002) (if the cell is an `NSPopUpButtonCell`), and then updates the datasource, if required. This method does not do any checks to see if the cell is enabled.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTableView.h`

preparedCellAtColumn:row:

Returns the fully prepared cell that the receiver will use for drawing or processing of the specified row and column.

```
- (NSCell *)preparedCellAtColumn:(NSInteger)column row:(NSInteger)row
```

Parameters

column
The column index for which to return the appropriate cell.

row
The row index for which to return the appropriate cell.

Return Value
New `NSCell` subclass instance to use for the specified *row* and *column*. The value for the cell is correctly set, and the delegate method `tableView:willDisplayCell:forTableColumn:row:` will have been called.

Discussion
You can override this method to do any additional cell set up that is required, or invoke it to retrieve a cell that has its contents configured for the specified *column* and *row*.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
Cocoa Tips and Tricks
DragNDropOutlineView
PhotoSearch

Declared In
`NSTableView.h`

rectOfColumn:

Returns the rectangle containing the column at a given index.

```
- (NSRect)rectOfColumn:(NSInteger)columnIndex
```

Parameters
columnIndex
The index of a column in the receiver.

Return Value
The rectangle containing the column at *columnIndex*. Returns `NSZeroRect` if *columnIndex* lies outside the range of valid column indices for the receiver.

Discussion
You can use this method to update a single column more efficiently than sending the table view a reloadData (page 2645) message.

```
[aTableView setNeedsDisplayInRect:[aTableView rectOfColumn:column]];
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `frameOfCellAtColumn:row:` (page 2635)
- `rectOfRow:` (page 2645)
- `headerRectOfColumn:` (page 2607) (`NSTableHeaderView`)

Declared In
`NSTableView.h`

rectOfRow:

Returns the rectangle containing the row at a given index.

```
- (NSRect)rectOfRow:(NSInteger)rowIndex
```

Return Value
The rectangle containing the row at *rowIndex*. Returns `NSZeroRect` if *rowIndex* lies outside the range of valid row indices for the receiver.

Discussion
You can use this method to update a single row more efficiently than sending the table view a reloadData (page 2645) message.

```
[aTableView setNeedsDisplayInRect:[aTableView rectOfRow:row]];
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `frameOfCellAtColumn:row:` (page 2635)
- `rectOfColumn:` (page 2644)

Related Sample Code
DragNDropOutlineView
NewsReader

Declared In
`NSTableView.h`

reloadData

Marks the receiver as needing redisplay, so it will reload the data for visible cells and draw the new values.

```
- (void)reloadData
```

Discussion
This method forces redraw of all the visible cells in the receiver. If you want to update the value in a single cell, column, or row, it is more efficient to use frameOfCellAtColumn:row: (page 2635), rectOfColumn: (page 2644), or rectOfRow: (page 2645) in conjunction with `setNeedsDisplayInRect:` (`NSView`). If you just want to update the scroller, use noteNumberOfRowsChanged (page 2641); if the height of a set of rows changes, use noteHeightOfRowsWithIndexesChanged: (page 2640).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `noteNumberOfRowsChanged` (page 2641)
- `noteHeightOfRowsWithIndexesChanged:` (page 2640)
- `frameOfCellAtColumn:row:` (page 2635)
- `rectOfColumn:` (page 2644)
- `rectOfRow:` (page 2645)

Related Sample Code
DragNDropOutlineView
QTAudioContextInsert
QTAudioExtractionPanel
WhackedTV
With and Without Bindings

Declared In
NSTableView.h

reloadDataForRowIndexes:columnIndexes:

Reloads the data for only the specified rows and columns.

```
- (void)reloadDataForRowIndexes:(NSIndexSet *)rowIndexes columnIndexes:(NSIndexSet *)columnIndexes
```

Parameters

rowIndexes
The indexes of the rows to update.

columnIndexes
The indexes of the columns to update.

Discussion
For cells that are visible, the appropriate `dataSource` (page 2627) and `delegate` (page 2628) methods will be called and the cells will be redrawn.

For tables that support variable row heights, the row height will not be re-queried from the delegate; it is your responsibility to invoke `noteHeightOfRowsWithIndexesChanged:` (page 2640) if a row height change is required.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
AnimatedTableView

Declared In
NSTableView.h

removeTableColumn:

Removes a given column from the receiver.

```
- (void)removeTableColumn:(NSTableColumn *)aTableColumn
```

Parameters

aTableColumn
The column to remove from the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sizeLastColumnToFit` (page 2669)
- `addTableColumn:` (page 2619)

Declared In
`NSTableView.h`

rowAtPoint:

Returns the index of the row a given point lies in.

```
- (NSInteger)rowAtPoint:(NSPoint)aPoint
```

Parameters

aPoint

A point in the coordinate system of the receiver.

Return Value
The index of the row *aPoint* lies in, or `-1` if *aPoint* lies outside the receiver's bounds.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `columnAtPoint:` (page 2625)

Declared In
`NSTableView.h`

rowHeight

Returns the height of each row in the receiver.

```
- (CGFloat)rowHeight
```

Return Value
The height of each row in the receiver.

Discussion
The default row height is `16.0`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setRowHeight:` (page 2666)

Related Sample Code
AnimatedTableView
Mountains
MP3 Player
PhotoSearch

Declared In
NSTableView.h

rowsInRect:

Returns a range of indices for the rows that lie wholly or partially within the vertical boundaries of a given rectangle.

```
- (NSRange)rowsInRect:(NSRect)aRect
```

Parameters

aRect

A rectangle in the coordinate system of the receiver.

Return Value
A range of indices for the receiver's rows that lie wholly or partially within the horizontal boundaries of *aRect*. If the width or height of *aRect* is 0, returns an NSRange whose length is 0.

Discussion
The location of the range is the first such row's index, and the length is the number of rows that lie in *aRect*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- columnsInRect: (page 2626)

Declared In
NSTableView.h

scrollColumnToVisible:

Scrolls the receiver and header view horizontally in an enclosing NSClipView so the column specified by *columnIndex* is visible.

```
- (void)scrollColumnToVisible:(NSInteger)columnIndex
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- scrollRowToVisible: (page 2648)
- scrollToPoint: (page 635) (NSClipView)

Declared In
NSTableView.h

scrollRowToVisible:

Scrolls the receiver vertically in an enclosing NSClipView so the row specified by *rowIndex* is visible.

```
- (void)scrollRowToVisible:(NSInteger)rowIndex
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `scrollColumnToVisible:` (page 2648)
- `scrollToPoint:` (page 635) (`NSClipView`)

Related Sample Code
DemoMonkey
MixMash

Declared In
`NSTableView.h`

selectAll:

Selects all rows or all columns, according to whether rows or columns were most recently selected.

```
- (void)selectAll:(id)sender
```

Parameters
sender
Typically the object that sent the message.

Discussion
If the table allows multiple selection, this action method selects all rows or all columns, according to whether rows or columns were most recently selected. If nothing has been recently selected, this method selects all rows. If this table doesn't allow multiple selection, this method does nothing.

If the selection does change, this method posts `NSTableViewSelectionDidChangeNotification` (page 2679) to the default notification center.

As a target-action method, `selectAll:` checks with the delegate before changing the selection.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsMultipleSelection` (page 2621)
- `deselectAll:` (page 2628)
- `selectColumn:byExtendingSelection:` (page 2649)

Declared In
`NSTableView.h`

selectColumn:byExtendingSelection:

Selects a column at a given index, optionally extending any existing selection. (Deprecated in Mac OS X v10.3. Use `selectColumnIndexes:byExtendingSelection:` (page 2650) instead.)

```
- (void)selectColumn:(NSInteger)columnIndex byExtendingSelection:(BOOL)flag
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

See Also
- `allowsMultipleSelection` (page 2621)
- `allowsColumnSelection` (page 2620)
- `deselectColumn:` (page 2629)
- `selectedColumn` (page 2650)
- `selectRow:byExtendingSelection:` (page 2653)

Declared In
`NSTableView.h`

selectColumnIndexes:byExtendingSelection:

Sets the column selection using *indexes*.

```
- (void)selectColumnIndexes:(NSIndexSet *)indexes byExtendingSelection:(BOOL)extend
```

Discussion
If the *extend* flag is `NO` the selected columns are specified by *indexes*. If *extend* is `YES`, the columns indicated by *indexes* are added to the collection of already selected columns, providing multiple selection.

If a subclass implements only the deprecated `selectColumn:byExtendingSelection:` (page 2649) method, then this method will be invoked in a loop. If a subclass implements this method, then `selectColumn:byExtendingSelection:` is not used. This allows subclasses that already implement `selectColumn:byExtendingSelection:` to still receive all selection messages. To avoid cycles, implementations of this method and `selectColumn:byExtendingSelection:` should not invoke each other.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `selectRowIndexes:byExtendingSelection:` (page 2654)

Declared In
`NSTableView.h`

selectedColumn

Returns the index of the last column selected or added to the selection.

```
- (NSInteger)selectedColumn
```

Return Value
The index of the last column selected or added to the selection, or `-1` if no column is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedColumnEnumerator` (page 2651)
- `numberOfSelectedColumns` (page 2642)
- `selectColumn:byExtendingSelection:` (page 2649)
- `deselectColumn:` (page 2629)

Declared In
`NSTableView.h`

selectedColumnEnumerator

This method has been deprecated. (Deprecated in Mac OS X v10.3. Use `selectedColumnIndexes` (page 2651) instead.)

```
- (NSEnumerator *)selectedColumnEnumerator
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

See Also
- `numberOfSelectedColumns` (page 2642)
- `selectedColumn` (page 2650)
- `selectedRowEnumerator` (page 2652)

Declared In
`NSTableView.h`

selectedColumnIndexes

Returns an index set containing the indexes of the selected columns.

```
- (NSIndexSet *)selectedColumnIndexes
```

Return Value
An index set containing the indexes of the selected columns.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `selectedRowIndexes` (page 2652)
- `selectColumnIndexes:byExtendingSelection:` (page 2650)

Declared In
`NSTableView.h`

selectedRow

Returns the index of the last row selected or added to the selection.

```
- (NSInteger)selectedRow
```

Return Value
The index of the last row selected or added to the selection, or `-1` if no row is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedRowEnumerator` (page 2652)
- `numberOfSelectedRows` (page 2642)
- `selectRow:byExtendingSelection:` (page 2653)
- `deselectRow:` (page 2630)

Related Sample Code
EnhancedAudioBurn
FunHouse
IdentitySample
QTAudioContextInsert
With and Without Bindings

Declared In
`NSTableView.h`

selectedRowEnumerator

This method has been deprecated. (Deprecated in Mac OS X v10.3. Use `selectedRowIndexes` (page 2652) instead.)

```
- (NSEnumerator *)selectedRowEnumerator
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

See Also
- `numberOfSelectedRows` (page 2642)
- `selectedRow` (page 2651)
- `selectedColumnEnumerator` (page 2651)

Declared In
`NSTableView.h`

selectedRowIndexes

Returns an index set containing the indexes of the selected rows.

```
- (NSIndexSet *)selectedRowIndexes
```

Return Value
An index set containing the indexes of the selected rows.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `selectedColumnIndexes` (page 2651)
- `selectRowIndexes:byExtendingSelection:` (page 2654)

Related Sample Code
AnimatedTableView
DragNDropOutlineView
IdentitySample
STUCAuthoringDeviceCocoaSample
With and Without Bindings

Declared In
`NSTableView.h`

selectionHighlightStyle

Returns the selection highlight style used by the receiver to indicate row and column selection.

```
- (NSTableViewSelectionHighlightStyle)selectionHighlightStyle
```

Return Value
The selection highlight style used by the receiver to use to indicate row and column selection. See "Selection Styles" (page 2678) for the possible values.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTableView.h`

selectRow:byExtendingSelection:

Selects a row at a given index, optionally extending any existing selection. (Deprecated in Mac OS X v10.3. Use `selectRowIndexes:byExtendingSelection:` (page 2654) instead.)

```
- (void)selectRow:(NSInteger)rowIndex byExtendingSelection:(BOOL)flag
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

See Also
- `allowsMultipleSelection` (page 2621)
- `deselectRow:` (page 2630)
- `selectedRow` (page 2651)
- `selectColumn:byExtendingSelection:` (page 2649)

Related Sample Code
EnhancedAudioBurn
EnhancedDataBurn
PDFKitLinker2
WhackedTV

Declared In
NSTableView.h

selectRowIndexes:byExtendingSelection:

Sets the row selection using *indexes*.

```
- (void)selectRowIndexes:(NSIndexSet *)indexes byExtendingSelection:(BOOL)extend
```

Discussion
If the *extend* flag is NO the selected rows are specified by *indexes*. If *extend* is YES, the rows indicated by *indexes* are added to the collection of already selected rows, providing multiple selection.

If a subclass implements only the deprecated selectRow:byExtendingSelection: (page 2653) method, then that method will be invoked in a loop. This allows subclasses that already implement selectRow:byExtendingSelection: to still receive all selection messages. If a subclass implements selectRowIndexes:byExtendingSelection:, then selectRow:byExtendingSelection: is not used. Note that to avoid cycles, implementations of this method and selectRow:byExtendingSelection: should not invoke each other.

Availability
Available in Mac OS X v10.3 and later.

See Also
- selectColumnIndexes:byExtendingSelection: (page 2650)

Related Sample Code
DemoMonkey
DragNDropOutlineView
IdentitySample

Declared In
NSTableView.h

setAllowsColumnReordering:

Controls whether the user can drag column headers to reorder columns.

```
- (void)setAllowsColumnReordering:(BOOL)flag
```

Parameters
flag
YES to allow the user to reorder columns, otherwise NO.

Discussion
The default is YES. You can rearrange columns programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `moveColumn:toColumn:` (page 2640)
- `allowsColumnReordering` (page 2619)

Declared In
`NSTableView.h`

setAllowsColumnResizing:

Controls whether the user can resize columns by dragging between headers.

```
- (void)setAllowsColumnResizing:(BOOL)flag
```

Parameters

flag

`YES` to allow the user to resize columns, otherwise `NO`.

Discussion
The default is `YES`. You can resize columns programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setWidth:` (page 2600) (`NSTableColumn`)
- `allowsColumnResizing` (page 2619)

Declared In
`NSTableView.h`

setAllowsColumnSelection:

Controls whether the user can select an entire column by clicking its header.

```
- (void)setAllowsColumnSelection:(BOOL)flag
```

Parameters

flag

`YES` to allow the user to select columns, otherwise `NO`.

Discussion
The default is `NO`. You can select columns programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectColumn:byExtendingSelection:` (page 2649)
- `setAllowsColumnReordering:` (page 2654)
- `allowsColumnSelection` (page 2620)

Declared In
NSTableView.h

setAllowsEmptySelection:

Controls whether the receiver allows zero rows or columns to be selected.

```
- (void)setAllowsEmptySelection:(BOOL)flag
```

Parameters

flag

YES if an empty selection is allowed, otherwise NO.

Discussion
The default is YES.

Unlike with the other settings that affect selection behavior, you cannot set an empty selection programmatically if empty selection is disallowed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- deselectAll: (page 2628)
- deselectColumn: (page 2629)
- deselectRow: (page 2630)
- allowsEmptySelection (page 2620)

Declared In
NSTableView.h

setAllowsMultipleSelection:

Controls whether the user can select more than one row or column at a time.

```
- (void)setAllowsMultipleSelection:(BOOL)flag
```

Parameters

flag

YES to allow the user to select multiple rows or columns, otherwise NO.

Discussion
The default is NO. You can select multiple columns or rows programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectColumn:byExtendingSelection: (page 2649)
- selectRow:byExtendingSelection: (page 2653)
- allowsMultipleSelection (page 2621)

Declared In
NSTableView.h

setAllowsTypeSelect:

Sets whether the receiver allows the user to type characters to select rows.

```
- (void)setAllowsTypeSelect:(BOOL)value
```

Parameters

value

YES if the receiver allows type selection, otherwise NO.

Availability
Available in Mac OS X v10.5 and later.

See Also
- allowsTypeSelect (page 2621)

Declared In
NSTableView.h

setAutoresizesAllColumnsToFit:

Controls whether the receiver proportionally resizes its columns to fit when its superview's frame changes. (Deprecated in Mac OS X v10.4. Use setColumnAutoresizingStyle: (page 2659) instead.)

```
- (void)setAutoresizesAllColumnsToFit:(BOOL)flag
```

Discussion
If *flag* is YES, the difference in width is distributed among the receiver's table columns; if *flag* is NO, only the last column is resized to fit.

To preserve compatibility this method sets the autoresizing style to NSTableViewUniformColumnAutoresizingStyle, if *flag* is YES. Otherwise the autoresizing style is set to NSTableViewLastColumnOnlyAutoresizingStyle.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

See Also
- setColumnAutoresizingStyle: (page 2659)

Declared In
NSTableView.h

setAutosaveName:

Sets the name under which table information is automatically saved to *name*.

```
- (void)setAutosaveName:(NSString *)name
```

Discussion
If *name* is different from the current name, this method also reads in the saved information and sets the order and width of this table view's columns to match.

The table information is saved separately for each user and for each application that user uses. Note that even though a table view has an autosave name, it may not be saving table information automatically. To set whether table information is being saved automatically, use `setAutosaveTableColumns:` (page 2658).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `autosaveName` (page 2622)
- `setAutosaveTableColumns:` (page 2658)

Declared In
`NSTableView.h`

setAutosaveTableColumns:

Sets whether the order and width of this table view's columns are automatically saved.

```
- (void)setAutosaveTableColumns:(BOOL)flag
```

Discussion
If *flag* is different from the current value, this method also reads in the saved information and sets the table options to match.

The table information is saved separately for each user and for each application that user uses. Note that if `autosaveName` (page 2622) returns `nil`, this setting is ignored and table information isn't saved.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `autosaveTableColumns` (page 2623)
- `setAutosaveName:` (page 2657)

Declared In
`NSTableView.h`

setBackgroundColor:

Sets the receiver's background color to a given color.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Parameters
aColor
The background color for the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setNeedsDisplay:` (page 3225) (`NSView`)
- `backgroundColor` (page 2623)

Declared In
`NSTableView.h`

setColumnAutoresizingStyle:

Sets the column autoresizing style of the receiver to a given style.

```
- (void)setColumnAutoresizingStyle:(NSTableViewColumnAutoresizingStyle)style
```

Parameters

style

The column autoresizing style for the receiver. For possible values, see "Autoresizing Styles" (page 2677).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `columnAutoresizingStyle` (page 2625)

Declared In
`NSTableView.h`

setCornerView:

Sets the receiver's corner view to a given view.

```
- (void)setCornerView:(NSView *)aView
```

Parameters

aView

The corner view for the receiver.

Discussion
The default corner view merely draws a bezeled rectangle using a blank `NSTableHeaderCell` object, but you can replace it with a custom view that displays an image or with a control that can handle mouse events, such as a select all button. Your custom corner view should be as wide as a vertical `NSScroller` object and as tall as the receiver's header view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHeaderView:` (page 2664)
- `cornerView` (page 2627)

Declared In
`NSTableView.h`

setDataSource:

Sets the receiver's data source to a given object.

```
- (void)setDataSource:(id < NSTableViewDataSource >)anObject
```

Parameters

anObject

The data source for the receiver. The object must implement the appropriate methods of the `NSTableDataSource` informal protocol.

Discussion

In a managed memory environment, the receiver maintains a weak reference to the data source (that is, it does not retain the data source, see Communicating With Objects). After setting the data source, this method invokes `tile` (page 2673).

This method raises an `NSInternalInconsistencyException` if *anObject* doesn't respond to either `numberOfRowsInTableView:` or `tableView:objectValueForTableColumn:row:`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- dataSource` (page 2627)

Related Sample Code

TimelineToTC

Declared In

`NSTableView.h`

setDelegate:

Sets the receiver's delegate to a given object.

```
- (void)setDelegate:(id < NSTableViewDelegate >)anObject
```

Parameters

anObject

The delegate for the receiver.

Discussion

In a managed memory environment, the receiver maintains a weak reference to the delegate (that is, it does not retain the delegate, see Communicating With Objects).

Special Considerations

When you call the tableview's `setDelegate:` (page 2660) method, the delegate is automatically registered for the following notifications with the following delegate methods:

- The notification named `NSTableViewSelectionDidChangeNotification` (page 2679) is configured to notify the delegate's `tableViewSelectionDidChange:` (page 3841).
- The notification named `NSTableViewSelectionDidChangeNotification` (page 2679) is configured to notify the delegate's `tableViewColumnDidMove:` (page 3840).

- The notification named `NSTableViewColumnDidResizeNotification` (page 2679) is configured to notify the delegate's `tableViewColumnDidResize:` (page 3841).
- The notification named `NSTableViewSelectionIsChangingNotification` (page 2679) is configured to notify the delegate's `tableViewSelectionIsChanging:` (page 3841).

Setting the delegate to nil will cause these notifications to be disconnected. Rather than setting the delegate to nil and listening for notifications (and expecting NSTableView to still function correctly) you should instead implement the appropriate delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `delegate` (page 2628)

Declared In
`NSTableView.h`

setDoubleAction:

Sets the message sent to the target when the user double-clicks an uneditable cell or a column header to a given selector.

```
- (void)setDoubleAction:(SEL)aSelector
```

Parameters

aSelector

The message the receiver sends to its target when the user double-clicks an uneditable cell or a column header.

Discussion
If the double-clicked cell is editable, this message isn't sent and the cell is edited instead. You can use this method to implement features such as sorting records according to the column that was double-clicked. See also `clickedRow` (page 2624) which you can use to determine if a row was clicked rather than the column heading.

For the method to have any effect, the receiver's action and target must be set to the class in which the selector is declared. See *Action Messages* for additional information on action messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAction:` (page 828) (`NSControl`)
- `setTarget:` (page 838) (`NSControl`)
- `doubleAction` (page 2630)

Related Sample Code
ScriptingBridgeFinder

Declared In
`NSTableView.h`

setDraggingDestinationFeedbackStyle:

Sets the feedback style displayed when the user drags over the table view.

```
- (void)setDraggingDestinationFeedbackStyle:(NSTableViewDraggingDestinationFeedbackStyle)style
```

Parameters

style

The dragging feedback style. See "NSTableViewDraggingDestinationFeedbackStyle" (page 2675) for the possible values.

Availability

Available in Mac OS X v10.6 and later.

See Also

- draggingDestinationFeedbackStyle (page 2631)

Declared In

NSTableView.h

setDraggingSourceOperationMask:forLocal:

Sets the default operation mask returned by `draggingSourceOperationMaskForLocal:` to *mask*.

```
- (void)setDraggingSourceOperationMask:(NSDragOperation)mask forLocal:(BOOL)isLocal
```

Discussion

If *isLocal* is `YES` then *mask* applies when the destination object is in the same application. If *isLocal* is `NO` then *mask* applies when the destination object in an application outside the receiver's application. NSTableView will archive the operation mask you set for each *isLocal* setting.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSTableView.h

setDrawsGrid:

Sets whether the receiver draws a grid. (Deprecated in Mac OS X v10.3. Use `setGridStyleMask:` (page 2664) instead.)

```
- (void)setDrawsGrid:(BOOL)flag
```

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Declared In

NSTableView.h

setDropRow:dropOperation:

Used if you wish to "retarget" the proposed drop.

```
- (void)setDropRow:(NSInteger)row dropOperation:(NSTableViewDropOperation)operation
```

Discussion
To specify a drop on the second row, one would specify *row* as 1, and *operation* as `NSTableViewDropOn`. To specify a drop below the last row, one would specify *row* as `[self numberOfRows]` and *operation* as `NSTableViewDropAbove`. Passing a value of `-1` for *row*, and `NSTableViewDropOn` as the *operation* causes the entire table view to be highlighted rather than a specific row. This is useful if the data displayed by the receiver does not allow the user to drop items at a specific row location.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DemoMonkey
EnhancedAudioBurn
iSpend
MP3 Player
With and Without Bindings

Declared In
`NSTableView.h`

setFocusedColumn:

Sets the currently focused column to the specified index.

```
- (void)setFocusedColumn:(NSInteger)focusedColumn
```

Parameters
focusedColumn
The index of the column to focus, or -1 if there should be no focused column.

Discussion
This method will redisplay the old previously `focusedColumn` (page 2635) and the newly `focusedColumn` (page 2635), if required.

The focused column has a focus ring drawn around the `selectedRow` (page 2651) that intersects with the `focusedColumn` (page 2635).

You should not override this method.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `focusedColumn` (page 2635)
- `shouldFocusCell:atColumn:row:` (page 2668)

Declared In
`NSTableView.h`

setGridColor:

Sets the color used to draw grid lines to a given color.

```
- (void)setGridColor:(NSColor *)aColor
```

Parameters

aColor

The color to use to draw grid lines.

Discussion

The default color is gray.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setDrawsGrid: (page 2662)
- drawGridInClipRect: (page 2632)
- gridColor (page 2636)

Declared In

NSTableView.h

setGridStyleMask:

Sets the grid style mask to specify if no grid lines, vertical grid lines, or horizontal grid lines should be displayed.

```
- (void)setGridStyleMask:(NSUInteger)gridType
```

Parameters

gridType

The grid style mask. Possible values for *gridType* are described in "Grid styles" (page 2676).

Availability

Available in Mac OS X v10.3 and later.

See Also

- gridStyleMask (page 2636)

Declared In

NSTableView.h

setHeaderView:

Sets the receiver's header view to a given header view.

```
- (void)setHeaderView:(NSTableHeaderView *)aHeaderView
```

Parameters

aHeaderView

The header view for the receiver.

Discussion
If *aHeaderView* is `nil`, the current header view is removed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setCornerView:` (page 2659)
- `headerView` (page 2637)

Declared In
`NSTableView.h`

setHighlightedTableColumn:

Sets *aTableColumn* to be the currently highlighted column header.

```
- (void)setHighlightedTableColumn:(NSTableColumn *)aTableColumn
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `highlightedTableColumn` (page 2637)

Related Sample Code
NSOperationSample

Declared In
`NSTableView.h`

setIndicatorImage:inTableColumn:

Sets the indicator image of *aTableColumn* to *anImage*.

```
- (void)setIndicatorImage:(NSImage *)anImage inTableColumn:(NSTableColumn
    *)aTableColumn
```

Discussion
anImage is retained and released by the table view as appropriate.

The default sorting order indicators are available as named `NSImage` objects. These images are accessed using `[NSImage imageNamed:]` passing either `@"NSAscendingSortIndicator"` (the "^" icon), and `@"NSDescendingSortIndicator"` (the "v" icon).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indicatorImageInTableColumn:` (page 2638)

Related Sample Code
NSOperationSample

Declared In
NSTableView.h

setIntercellSpacing:

Sets the width and height between cells to those in a given NSSize struct.

```
- (void)setIntercellSpacing:(NSSize)aSize
```

Parameters

aSize

An NSSize struct that defines the width and height between cells in the receiver.

Discussion
The receiver redisplays after the new value is set.

The default intercell spacing is (3.0, 2.0).

Table views normally have a 1 pixel separation between consecutively selected rows or columns. An intercell spacing of (1.0, 1.0) or greater is required if you want this separation. An intercell spacing of (0.0, 0.0) forces there to be no separation between consecutive selections.

Availability
Available in Mac OS X v10.0 and later.

See Also
- intercellSpacing (page 2638)

Declared In
NSTableView.h

setRowHeight:

Sets the height for rows to a given value.

```
- (void)setRowHeight:(CGFloat)rowHeight
```

Parameters

rowHeight

The height for rows.

Discussion
After the height is set, this method invokes tile (page 2673).

Availability
Available in Mac OS X v10.0 and later.

See Also
- rowHeight (page 2647)

Related Sample Code
Mountains

Declared In
NSTableView.h

setSelectionHighlightStyle:

Sets the selection highlight style used by the receiver to indicate row and column selection.

```
- (void)setSelectionHighlightStyle:(NSTableViewSelectionHighlightStyle)selectionHighlightStyle
```

Parameters

selectionHighlightStyle

The selection highlight style to use to indicate row and column selection. See "Selection Styles" (page 2678) for the possible values.

Discussion
Setting the selection highlight style to `NSTableViewSelectionHighlightStyleSourceList` (page 2678) causes the receiver to draw its background using the source list style.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTableView.h

setSortDescriptors:

Sets the receiver's sort descriptors to the `NSSortDescriptor` objects in *array*.

```
- (void)setSortDescriptors:(NSArray *)array
```

Discussion
A table column is considered sortable if it has a sort descriptor that specifies the sorting direction, a key to sort by, and a selector defining how to sort. The array of sort descriptors is archived. Sort descriptors persist along with other column information if an autosave name is set.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `sortDescriptors` (page 2670)

Declared In
NSTableView.h

setUsesAlternatingRowBackgroundColors:

Sets whether the receiver uses the standard alternating row colors for its background.

```
- (void)setUsesAlternatingRowBackgroundColors:(BOOL)useAlternatingRowColors
```

Parameters

useAlternatingRowColors

YES to specify standard alternating row colors for the background, NO to specify a solid color.

Availability
Available in Mac OS X v10.3 and later.

See Also
- usesAlternatingRowBackgroundColors (page 2673)

Declared In
NSTableView.h

setVerticalMotionCanBeginDrag:

Sets whether vertical motion is treated as a drag or selection change to *flag*.

```
- (void)setVerticalMotionCanBeginDrag:(BOOL)flag
```

Discussion
If *flag* is NO then vertical motion will not start a drag. The default is YES.

Note that horizontal motion is always a valid motion to begin a drag. Most often, you would want to disable vertical dragging when it's expected that horizontal dragging is the natural motion.

Availability
Available in Mac OS X v10.0 and later.

See Also
- verticalMotionCanBeginDrag (page 2674)

Related Sample Code
QTKitMovieShuffler

Declared In
NSTableView.h

shouldFocusCell:atColumn:row:

Returns whether the fully cell at the specified row and column can be made the focused cell or not.

```
- (BOOL)shouldFocusCell:(NSCell *)cell atColumn:(NSInteger)column row:(NSInteger)row
```

Parameters

cell

The prepared cell to be focused upon.

column

The column of the cell.

row

The row of the cell.

Return Value
YES if the cell can be made the focused cell, otherwise NO.

Discussion
By default, only cells that are enabled can be focused. In addition, if the cell is an `NSTextFieldCell`, it will can only be focused if it is selectable or editable, and the table view delegate responds `YES` to `-tableView:shouldEditTableColumn:row:` (page 3834). Subclasses can override this to further control what cells can and cannot be made focused.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `focusedColumn` (page 2635)
- `setFocusedColumn:` (page 2663)

Declared In
`NSTableView.h`

sizeLastColumnToFit

Resizes the last column if there's room so the receiver fits exactly within its enclosing clip view.

```
- (void)sizeLastColumnToFit
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAutoresizesAllColumnsToFit:` (page 2657)
- `minWidth` (page 2594) (`NSTableColumn`)
- `maxWidth` (page 2593) (`NSTableColumn`)

Declared In
`NSTableView.h`

sizeToFit

Changes the width of columns in the receiver so all columns are visible.

```
- (void)sizeToFit
```

Discussion
All columns are resized to the same size, up to a column's maximum size. This method then invokes `tile` (page 2673).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSTableView.h`

sortDescriptors

Returns the receiver's sort descriptors.

```
- (NSArray *)sortDescriptors
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- setSortDescriptors: (page 2667)

Related Sample Code
DictionaryController

Declared In
NSTableView.h

tableColumns

Returns an array containing the the NSTableColumn objects in the receiver.

```
- (NSArray *)tableColumns
```

Return Value
An array containing the the NSTableColumn objects in the receiver.

Discussion
The array returned by tableColumns contains all receiver's columns, including those that are hidden.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AnimatedTableView
DragNDropOutlineView
NSOperationSample
People

Declared In
NSTableView.h

tableColumnWithIdentifier:

Returns the NSTableColumn object for the first column whose identifier is equal to a given object.

```
- (NSTableColumn *)tableColumnWithIdentifier:(id)anObject
```

Parameters
anObject
A column identifier.

Return Value
The `NSTableColumn` object for the first column whose identifier is equal to *`anObject`*, as compared using `isEqual:`, or `nil` if no columns are found with the specified identifier.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- columnWithIdentifier:` (page 2627)

Related Sample Code
DragNDropOutlineView
QTKitMovieShuffler

Declared In
`NSTableView.h`

textDidBeginEditing:

Posts an `NSControlTextDidBeginEditingNotification` (page 847) to the default notification center.

```
- (void)textDidBeginEditing:(NSNotification *)aNotification
```

Parameters
`aNotification`
The notification posted by the field editor; see the `NSText` class specifications for more information on this text delegate method.

Discussion
For more details, see the `NSControl` class specification.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- textShouldBeginEditing:` (page 2672)

Declared In
`NSTableView.h`

textDidChange:

Sends `textDidChange:` (page 2671) to the edited cell and posts an `NSControlTextDidChangeNotification` (page 848) to the default notification center.

```
- (void)textDidChange:(NSNotification *)aNotification
```

Parameters
`aNotification`
The notification posted by the field editor.

Discussion
See the `NSText` class specification for more information on this text delegate method. For additional details, see the `NSControl` class specification.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

textDidEndEditing:

Updates the data source based on the newly edited value and selects another cell for editing if possible according to the character that ended editing (Return, Tab, Backtab).

```
- (void)textDidEndEditing:(NSNotification *)aNotification
```

Discussion
`aNotification` is the NSNotification posted by the field editor; see the `NSText` class specification for more information on this text delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- textShouldEndEditing:` (page 2673)

Declared In
`NSTableView.h`

textShouldBeginEditing:

Queries the delegate using `control:textShouldBeginEditing:`, returning the delegate's response, or simply returning `YES` to allow editing of *`textObject`* if the delegate doesn't respond to that method.

```
- (BOOL)textShouldBeginEditing:(NSText *)textObject
```

Discussion
See the `NSText` class specification for more information on this text delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- textDidBeginEditing:` (page 2671)

Declared In
`NSTableView.h`

textShouldEndEditing:

Validates the *textObject* cell being edited and queries the delegate using `control:textShouldEndEditing:`, returning the delegate's response if it responds to that method.

```
- (BOOL)textShouldEndEditing:(NSText *)textObject
```

Discussion
If it doesn't, it returns `YES` if the cell's new value is valid and `NO` if it isn't. See the `NSText` class specification for more information on this text delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- textDidEndEditing:` (page 2672)

Declared In
`NSTableView.h`

tile

Properly sizes the receiver and its header view and marks it as needing display.

```
- (void)tile
```

Discussion
Also resets cursor rectangles for the header view and line scroll amounts for the `NSScrollView` object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setNeedsDisplay:` (page 3225) (`NSView`)

Declared In
`NSTableView.h`

usesAlternatingRowBackgroundColors

Returns a Boolean value that indicates whether the receiver uses the standard alternating row colors for its background.

```
- (BOOL)usesAlternatingRowBackgroundColors
```

Return Value
`YES` if the receiver uses standard alternating row colors for the background, `NO` if it uses a solid color.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setUsesAlternatingRowBackgroundColors:` (page 2667)

Declared In
NSTableView.h

verticalMotionCanBeginDrag

Returns whether vertical motion is treated as a drag or selection change.

```
- (BOOL)verticalMotionCanBeginDrag
```

Discussion
NO means that vertical motion will not start a drag. Note that horizontal motion is always a valid motion to begin a drag.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setVerticalMotionCanBeginDrag: (page 2668)

Declared In
NSTableView.h

Delegate Methods

tableView:writeRows:toPasteboard:

Writes the specified rows to the specified pasteboard. (Deprecated in Mac OS X v10.4. This method has been deprecated. You should implement tableView:writeRowsWithIndexes:toPasteboard: (page 3825) instead.)

```
- (BOOL)tableView:(NSTableView *)aTableView writeRows:(NSArray *)rows
    toPasteboard:(NSPasteboard *)pboard
```

Discussion
Invoked by *aTableView* after it has been determined that a drag should begin, but before the drag has been started. To refuse the drag, return NO. To start a drag, return YES and place the drag data onto *pboard* (data, owner, and so on). The drag image and other drag-related information will be set up and provided by the table view once this call returns with YES. *rows* is the list of row numbers that will be participating in the drag.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
NSTableView.h

Constants

NSTableViewDraggingDestinationFeedbackStyle

These constants specify the drag styles displayed by the table view. They're used by the `draggingDestinationFeedbackStyle` (page 2631) and `setDraggingDestinationFeedbackStyle:` (page 2662)

```
enum {
    NSTableViewDraggingDestinationFeedbackStyleNone = -1,
    NSTableViewDraggingDestinationFeedbackStyleRegular = 0,
    NSTableViewDraggingDestinationFeedbackStyleSourceList = 1,
};
#endif
typedef NSInteger NSTableViewDraggingDestinationFeedbackStyle;
```

Constants

`NSTableViewDraggingDestinationFeedbackStyleNone`

Provides no feedback when the user drags over the table view. This option exists to allow subclasses to implement their dragging destination highlighting, or to make it not show anything all.

Available in Mac OS X v10.6 and later.

Declared in `NSTableView.h`.

`NSTableViewDraggingDestinationFeedbackStyleRegular`

Draws a solid round-rect background on drop target rows, and an insertion marker between rows. This style should be used in most cases.

Available in Mac OS X v10.6 and later.

Declared in `NSTableView.h`.

`NSTableViewDraggingDestinationFeedbackStyleSourceList`

Draws an outline on drop target rows, and an insertion marker between rows. This style will automatically be set for source lists when the table's `setSelectionHighlightStyle:` (page 2667) is set to `NSTableViewSelectionHighlightStyleSourceList` (page 2678). This is the standard look for Source Lists, but may be used in other areas as needed.

Available in Mac OS X v10.6 and later.

Declared in `NSTableView.h`.

Drop Operations

`NSTableView` defines these constants to specify drop operations.

```
enum {
    NSTableViewDropOn,
    NSTableViewDropAbove
};
typedef NSUInteger NSTableViewDropOperation;
```

Constants

`NSTableViewDropOn`

Specifies that the drop should occur on the specified row.

Available in Mac OS X v10.0 and later.

Declared in `NSTableView.h`.

`NSTableViewDropAbove`

Specifies that the drop should occur above the specified row.

Available in Mac OS X v10.0 and later.

Declared in `NSTableView.h`.

Discussion

For example, given a table with `n` rows (numbered with row `0` at the top visually), a row of `n-1` and operation of `NSTableViewDropOn` would specify a drop on the last row. To specify a drop below the last row, you use a row of `n` and `NSTableViewDropAbove` for the operation.

Declared In

`NSTableView.h`

Grid styles

`NSTableView` defines these constants to specify grid styles. These constants are used by `gridStyleMask` (page 2636) and `setGridStyleMask:` (page 2664). The mask can be either `NSTableViewGridNone` (page 2676) or it can contain either or both of the other options combined using the C bitwise `OR` operator.

```
enum {
    NSTableViewGridNone                    = 0,
    NSTableViewSolidVerticalGridLineMask   = 1 << 0,
    NSTableViewSolidHorizontalGridLineMask = 1 << 1
};
```

Constants

`NSTableViewGridNone`

Specifies that no grid lines should be displayed.

Available in Mac OS X v10.3 and later.

Declared in `NSTableView.h`.

`NSTableViewSolidVerticalGridLineMask`

Specifies that vertical grid lines should be displayed.

Available in Mac OS X v10.3 and later.

Declared in `NSTableView.h`.

`NSTableViewSolidHorizontalGridLineMask`

Specifies that horizontal grid lines should be displayed.

Available in Mac OS X v10.3 and later.

Declared in `NSTableView.h`.

Autoresizing Styles

The following constants specify the autoresizing styles. These constants are used by `columnAutoresizingStyle` (page 2625) and `setColumnAutoresizingStyle:` (page 2659).

```
enum {
    NSTableViewNoColumnAutoresizing = 0,
    NSTableViewUniformColumnAutoresizingStyle,
    NSTableViewSequentialColumnAutoresizingStyle,
    NSTableViewReverseSequentialColumnAutoresizingStyle,
    NSTableViewLastColumnOnlyAutoresizingStyle,
    NSTableViewFirstColumnOnlyAutoresizingStyle
};
typedef NSUInteger NSTableViewColumnAutoresizingStyle;
```

Constants

`NSTableViewNoColumnAutoresizing`
Disable table column autoresizing.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

`NSTableViewUniformColumnAutoresizingStyle`
Autoresize all columns by distributing space equally, simultaneously.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

`NSTableViewSequentialColumnAutoresizingStyle`
Autoresize each table column sequentially, from the last auto-resizable column to the first auto-resizable column; proceed to the next column when the current column has reached its minimum or maximum size.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

`NSTableViewReverseSequentialColumnAutoresizingStyle`
Autoresize each table column sequentially, from the first auto-resizable column to the last auto-resizable column; proceed to the next column when the current column has reached its minimum or maximum size.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

`NSTableViewLastColumnOnlyAutoresizingStyle`
Autoresize only the last table column.

When that table column can no longer be resized, stop autoresizing. Normally you should use one of the sequential autoresizing modes instead.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

`NSTableViewFirstColumnOnlyAutoresizingStyle`
Autoresize only the first table column.

When that table column can no longer be resized, stop autoresizing. Normally you should use one of the sequential autoresizing modes instead.

Available in Mac OS X v10.4 and later.

Declared in `NSTableView.h`.

Selection Styles

The following constants specify the selection highlight styles. These constants are used by `selectionHighlightStyle` (page 2653) and `setSelectionHighlightStyle:` (page 2667).

```
enum {
    NSTableViewSelectionHighlightStyleNone = -1,
    NSTableViewSelectionHighlightStyleRegular = 0,
    NSTableViewSelectionHighlightStyleSourceList = 1,
};
typedef NSInteger NSTableViewSelectionHighlightStyle;
```

Constants

`NSTableViewSelectionHighlightStyleNone`
Displays no highlight style at all.

Available in Mac OS X v10.6 and later.

Declared in `NSTableView.h`.

`NSTableViewSelectionHighlightStyleRegular`
The regular highlight style of NSTableView. On Mac OS X v10.5 a light blue (returned by sending `NSColor` a `alternateSelectedControlColor` (page 666) message) or light gray color (returned by sending NSColor a `secondarySelectedControlColor` (page 688) message).

Available in Mac OS X v10.5 and later.

Declared in `NSTableView.h`.

`NSTableViewSelectionHighlightStyleSourceList`
The source list style of NSTableView. On 10.5, a light blue gradient is used to highlight selected rows.

Note: When using this style, cell subclasses that implement `drawsBackground` must set the value to `NO`. Otherwise, the cells will draw over the tableview's highlighting.

Available in Mac OS X v10.5 and later.

Declared in `NSTableView.h`.

Notifications

NSTableViewColumnDidMoveNotification

Posted whenever a column is moved by user action in an `NSTableView` object. The notification object is the table view in which a column moved. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSOldColumn"`	An `NSNumber` object containing the integer value of the column's original index.
`@"NSNewColumn"`	An `NSNumber` object containing the integer value of the column's present index.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- moveColumn:toColumn:` (page 2640)

Declared In
`NSTableView.h`

NSTableViewColumnDidResizeNotification

Posted whenever a column is resized in an `NSTableView` object. The notification object is the table view in which a column was resized. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSTableColumn"`	The column that was resized.
`@"NSOldWidth"`	An NSNumber containing the integer value of the column's original width.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

NSTableViewSelectionDidChangeNotification

Posted after an `NSTableView` object's selection changes. The notification object is the table view whose selection changed. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTableView.h`

NSTableViewSelectionIsChangingNotification

Posted as an `NSTableView` object's selection changes (while the mouse button is still down). The notification object is the table view whose selection is changing. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

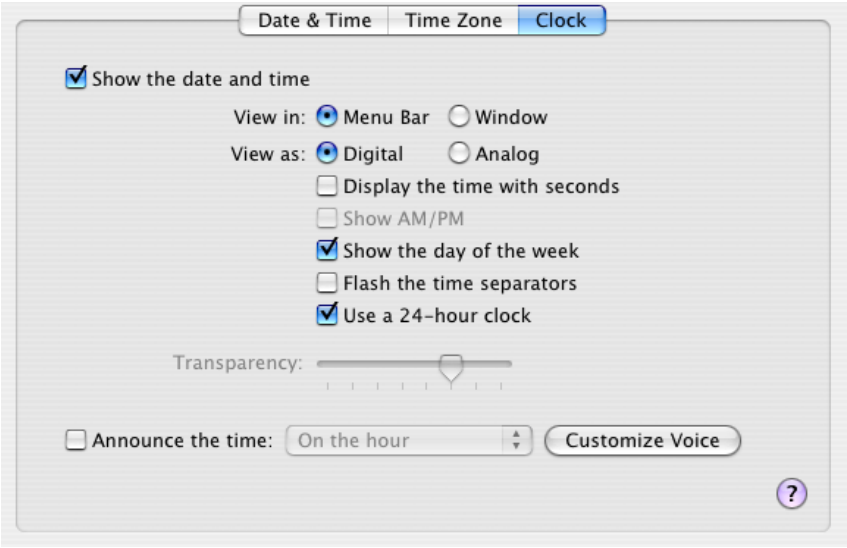
Declared In
`NSTableView.h`

NSTabView Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTabView.h
Companion guide	Tab Views
Related sample code	EnhancedAudioBurn iChatTheater MyPhoto PDF Annotation Editor Reducer

Overview

An `NSTabView` object provides a convenient way to present information in multiple pages. The view contains a row of tabs that give the appearance of folder tabs, as shown in the following figure. The user selects the desired page by clicking the appropriate tab or using the arrow keys to move between pages. Each page displays a view hierarchy provided by your application.



Tasks

Adding and Removing Tabs

- `addTabViewItem:` (page 2685)
  Adds the tab item specified by *`tabViewItem`*.
- `insertTabViewItem:atIndex:` (page 2689)
  Inserts *`tabViewItem`* into the receiver's array of tab view items at *`index`*.
- `removeTabViewItem:` (page 2690)
  Removes the item specified by *`tabViewItem`* from the receiver's array of tab view items.

Accessing Tabs

- `indexOfTabViewItem:` (page 2688)
  Returns the index of the specified item in the tab view.
- `indexOfTabViewItemWithIdentifier:` (page 2688)
  Returns the index of the item that matches the specified identifier. *`identifier`*, or `NSNotFound` if the item is not found.
- `numberOfTabViewItems` (page 2690)
  Returns the number of items in the receiver's array of tab view items.
- `tabViewItemAtIndex:` (page 2697)
  Returns the tab view item at *`index`* in the tab view's array of items.
- `tabViewItems` (page 2698)
  Returns the receiver's array of tab view items.

Selecting a Tab

- `selectFirstTabViewItem:` (page 2691)
  This action method selects the first tab view item.
- `selectLastTabViewItem:` (page 2691)
  This action method selects the last tab view item.
- `selectNextTabViewItem:` (page 2692)
  This action method selects the next tab view item in the sequence.
- `selectPreviousTabViewItem:` (page 2692)
  This action method selects the previous tab view item in the sequence.
- `selectTabViewItem:` (page 2693)
  Selects the specified tab view item.
- `selectTabViewItemAtIndex:` (page 2693)
  Selects the tab view item specified by *index*.
- `selectTabViewItemWithIdentifier:` (page 2694)
  Selects the tab view item specified by *identifier*.
- `selectedTabViewItem` (page 2691)
  Returns the tab view item for the currently selected tab
- `takeSelectedTabViewItemFromSender:` (page 2698)
  Sets the selected tab view item to the selected item obtained from the sender.

Modifying the Font

- `font` (page 2687)
  Returns the font for tab label text.
- `setFont:` (page 2696)
  Sets the font for tab label text to *font*.

Modifying the Tab Type

- `setTabViewType:` (page 2697)
  Sets the tab type to *tabViewType*.
- `tabViewType` (page 2698)
  Returns the tab type for the receiver.

Modifying Controls Tint

- `controlTint` (page 2686)
  Returns the receiver's control tint.
- `setControlTint:` (page 2695)
  Sets the receiver's control tint to *controlTint*.

Manipulating the Background

- `drawsBackground` (page 2687)
  Returns whether if the receiver draws a background color when the tab view type is `NSNoTabsNoBorder`.
- `setDrawsBackground:` (page 2696)
  Sets whether a background is drawn when the view type is `NSNoTabsNoBorder` to *`flag`*.

Determining the Size

- `minimumSize` (page 2689)
  Returns the minimum size necessary for the receiver to display tabs in a useful way.
- `contentRect` (page 2686)
  Returns the rectangle describing the content area of the receiver.
- `controlSize` (page 2686)
  Returns the size of the receiver.
- `setControlSize:` (page 2695)
  Sets the size of the receiver to *`controlSize`*.

Truncating Tab Labels

- `allowsTruncatedLabels` (page 2685)
  Returns whether if the receiver allows truncating for labels that don't fit on a tab.
- `setAllowsTruncatedLabels:` (page 2694)
  Sets whether the receiver allows truncating for names that don't fit on a tab.

Assigning a Delegate

- `setDelegate:` (page 2695)
  Sets the receiver's delegate.
- `delegate` (page 2687)
  Returns the receiver's delegate.

Event Handling

- `tabViewItemAtPoint:` (page 2697)
  Returns the tab view item at the specified point.

Instance Methods

addTabViewItem:

Adds the tab item specified by *tabViewItem*.

```
- (void)addTabViewItem:(NSTabViewItem *)tabViewItem
```

Parameters

tabViewItem

The tab view item to be added.

Discussion

The item is added at the end of the array of tab items, so the new tab appears on the right side of the view. If the delegate supports it, invokes the delegate's `tabViewDidChangeNumberOfTabViewItems:` (page 3845) method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `insertTabViewItem:atIndex:` (page 2689)
- `numberOfTabViewItems` (page 2690)
- `removeTabViewItem:` (page 2690)
- `tabViewItemAtIndex:` (page 2697)
- `tabViewItems` (page 2698)

Declared In

`NSTabView.h`

allowsTruncatedLabels

Returns whether if the receiver allows truncating for labels that don't fit on a tab.

```
- (BOOL)allowsTruncatedLabels
```

Return Value

`YES` if the receiver allows truncating for labels that don't fit on a tab, otherwise `NO`.

Discussion

The default is `NO`.

When truncating is allowed, the tab view inserts an ellipsis, if necessary, to fit a label in the tab.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setAllowsTruncatedLabels:` (page 2694)

Declared In

`NSTabView.h`

contentRect

Returns the rectangle describing the content area of the receiver.

```
- (NSRect)contentRect
```

Return Value
Returns the rectangle describing the content area of the receiver.

Discussion
This area does not include the space required for the receiver's tabs or borders (if any).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTabView.h`

controlSize

Returns the size of the receiver.

```
- (NSControlSize)controlSize
```

Return Value
Valid return values are described in`Control Sizes` in *NSCell Class Reference*.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setControlSize:` (page 2695)

Declared In
`NSTabView.h`

controlTint

Returns the receiver's control tint.

```
- (NSControlTint)controlTint
```

Return Value
The tab view's control tint. See `NSControlTint` (page 619) for the supported values.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setControlTint:` (page 2695)

Declared In
`NSTabView.h`

delegate

Returns the receiver's delegate.

```
- (id < NSTabViewDelegate >)delegate
```

Return Value
The object's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDelegate:` (page 2695)

Declared In
`NSTabView.h`

drawsBackground

Returns whether if the receiver draws a background color when the tab view type is `NSNoTabsNoBorder`.

```
- (BOOL)drawsBackground
```

Return Value
`YES` if the receiver draws a background color when the tab view type is `NSNoTabsNoBorder`, otherwise `NO`.

Discussion
If the receiver uses bezeled edges or a line border, the appropriate background color for that border is used.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTabViewType:` (page 2697)
- `setDrawsBackground:` (page 2696)

Declared In
`NSTabView.h`

font

Returns the font for tab label text.

```
- (NSFont *)font
```

Return Value
Returns the tab view label font.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFont:` (page 2696)

Declared In
NSTabView.h

indexOfTabViewItem:

Returns the index of the specified item in the tab view.

```
- (NSInteger)indexOfTabViewItem:(NSTabViewItem *)tabViewItem
```

Parameters

tabViewItem
The tab view item.

Return Value
The zero-based index of *tabViewItem*, or [NSNotFound] if the item is not found.

Discussion
The returned index is zero-based.

Availability
Available in Mac OS X v10.0 and later.

See Also
- indexOfTabViewItemWithIdentifier: (page 2688)
- insertTabViewItem:atIndex: (page 2689)
- numberOfTabViewItems (page 2690)
- tabViewItemAtIndex: (page 2697)

Declared In
NSTabView.h

indexOfTabViewItemWithIdentifier:

Returns the index of the item that matches the specified identifier. *identifier*, or NSNotFound if the item is not found.

```
- (NSInteger)indexOfTabViewItemWithIdentifier:(id)identifier
```

Parameters

identifier
The identifier of a tab view item.

Return Value
The zero-based index of the tab view item corresponding to *identifier*, or [NSNotFound] if the item is not found.

Discussion
The returned index is zero-based.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfTabViewItem:` (page 2688)
- `insertTabViewItem:atIndex:` (page 2689)
- `numberOfTabViewItems` (page 2690)
- `tabViewItemAtIndex:` (page 2697)

Declared In
`NSTabView.h`

insertTabViewItem:atIndex:

Inserts *tabViewItem* into the receiver's array of tab view items at *index*.

```
- (void)insertTabViewItem:(NSTabViewItem *)tabViewItem atIndex:(NSInteger)index
```

Parameters

tabViewItem
The tab view item to be added.

index
The index at which to insert the tab view item. The *index* parameter is zero-based.

Discussion
If there is a delegate and the delegate supports it, sends the delegate the `tabViewDidChangeNumberOfTabViewItems:` (page 3845) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfTabViewItem:` (page 2688)
- `indexOfTabViewItemWithIdentifier:` (page 2688)
- `numberOfTabViewItems` (page 2690)
- `tabViewItemAtIndex:` (page 2697)

Declared In
`NSTabView.h`

minimumSize

Returns the minimum size necessary for the receiver to display tabs in a useful way.

```
- (NSSize)minimumSize
```

Return Value
The minimum size necessary for the receiver to display tabs in a useful way.

Discussion
You can use the value returned by this method to limit how much a user can resize a tab view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTabViewType:` (page 2697)

Declared In
`NSTabView.h`

numberOfTabViewItems

Returns the number of items in the receiver's array of tab view items.

```
- (NSInteger)numberOfTabViewItems
```

Return Value
The number of items in the tab view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfTabViewItem:` (page 2688)
- `tabViewItems` (page 2698)

Declared In
`NSTabView.h`

removeTabViewItem:

Removes the item specified by *tabViewItem* from the receiver's array of tab view items.

```
- (void)removeTabViewItem:(NSTabViewItem *)tabViewItem
```

Parameters
tabViewItem
The tab view item to be removed.

Discussion
If there is a delegate and the delegate supports it, sends the delegate the `tabViewDidChangeNumberOfTabViewItems:` (page 3845) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addTabViewItem:` (page 2685)
- `insertTabViewItem:atIndex:` (page 2689)
- `tabViewItems` (page 2698)

Declared In
`NSTabView.h`

selectedTabViewItem

Returns the tab view item for the currently selected tab

```
- (NSTabViewItem *)selectedTabViewItem
```

Return Value
The currently selected tab view item, or `nil` if no item is selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- selectTabViewItemAtIndex:` (page 2693)

Related Sample Code
iChatTheater

Declared In
`NSTabView.h`

selectFirstTabViewItem:

This action method selects the first tab view item.

```
- (void)selectFirstTabViewItem:(id)sender
```

Parameters

sender
Typically the object that sent invoked the message.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- selectTabViewItem:` (page 2693)

Declared In
`NSTabView.h`

selectLastTabViewItem:

This action method selects the last tab view item.

```
- (void)selectLastTabViewItem:(id)sender
```

Parameters

sender
Typically the object that sent invoked the message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectTabViewItem: (page 2693)

Declared In
NSTabView.h

selectNextTabViewItem:

This action method selects the next tab view item in the sequence.

```
- (void)selectNextTabViewItem:(id)sender
```

Parameters

sender

Typically the object that sent invoked the message.

Discussion
If the currently visible item is the last item in the sequence, this method does nothing, and the last page remains displayed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectTabViewItem: (page 2693)

Declared In
NSTabView.h

selectPreviousTabViewItem:

This action method selects the previous tab view item in the sequence.

```
- (void)selectPreviousTabViewItem:(id)sender
```

Parameters

sender

Typically the object that sent invoked the message.

Discussion
If the currently visible item is the first item in the sequence, this method does nothing, and the first page remains displayed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectTabViewItem: (page 2693)

Declared In
NSTabView.h

selectTabViewItem:

Selects the specified tab view item.

```
- (void)selectTabViewItem:(NSTabViewItem *)tabViewItem
```

Parameters

tabViewItem

The tab item to select.

Discussion

If there is a delegate and the delegate supports it, sends the delegate the `tabView:shouldSelectTabViewItem:` (page 3844) message.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- insertTabViewItem:atIndex:` (page 2689)

`- selectedTabViewItem` (page 2691)

Related Sample Code

From A View to A Movie

From A View to A Picture

iChatTheater

Declared In

`NSTabView.h`

selectTabViewItemAtIndex:

Selects the tab view item specified by *index*.

```
- (void)selectTabViewItemAtIndex:(NSInteger)index
```

Parameters

index

The index of the tab item to selected.

Discussion

The *index* parameter is base 0. If there is a delegate and the delegate supports it, sends the delegate the `tabView:shouldSelectTabViewItem:` (page 3844) message.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- insertTabViewItem:atIndex:` (page 2689)

`- selectedTabViewItem` (page 2691)

Related Sample Code

EnhancedAudioBurn

PDFKitLinker2

QTAudioContextInsert

XMLBrowser

Declared In
NSTabView.h

selectTabViewItemWithIdentifier:

Selects the tab view item specified by *identifier*.

```
- (void)selectTabViewItemWithIdentifier:(id)identifier
```

Parameters

identifier
The identifier of the tab item to select.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setIdentifier: (page 2705) (NSTabViewItem)
- identifier (page 2704) (NSTabViewItem)
- selectTabViewItemAtIndex: (page 2693)
- selectedTabViewItem (page 2691)

Related Sample Code
FinalCutPro_AppleEvents

Declared In
NSTabView.h

setAllowsTruncatedLabels:

Sets whether the receiver allows truncating for names that don't fit on a tab.

```
- (void)setAllowsTruncatedLabels:(BOOL)allowTruncatedLabels
```

Parameters

allowTruncatedLabels
YES if the receiver allows truncating for labels that don't fit on a tab, otherwise NO.

Discussion
The default is NO.

When truncating is allowed, the tab view inserts an ellipsis, if necessary, to fit a label in the tab.

Availability
Available in Mac OS X v10.0 and later.

See Also
- allowsTruncatedLabels (page 2685)

Declared In
NSTabView.h

setControlSize:

Sets the size of the receiver to *controlSize*.

```
- (void)setControlSize:(NSControlSize)controlSize
```

Parameters

controlSize

The size of the receiver. Valid values are described in `Control Sizes` in *NSCell Class Reference*

Availability
Available in Mac OS X v10.0 and later.

See Also
- `controlSize` (page 2686)

Declared In
`NSTabView.h`

setControlTint:

Sets the receiver's control tint to *controlTint*.

```
- (void)setControlTint:(NSControlTint)controlTint
```

Parameters

controlTint

The tab view's control tint. See `NSControlTint` (page 619) for the supported values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `controlTint` (page 2686)

Declared In
`NSTabView.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSTabViewDelegate >)anObject
```

Parameters

anObject

The delegate object. It must conform to the `NSTabViewDelegate` protocol.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `delegate` (page 2687)

Related Sample Code
JSPong

Declared In
NSTabView.h

setDrawsBackground:

Sets whether a background is drawn when the view type is NSNoTabsNoBorder to *flag*.

```
- (void)setDrawsBackground:(BOOL)flag
```

Parameters

flag

YES if the background should be drawn, otherwise NO.

Discussion
If the receiver has a bezeled border or a line border, the appropriate background for that border is used.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTabViewType: (page 2697)
- drawsBackground (page 2687)

Declared In
NSTabView.h

setFont:

Sets the font for tab label text to *font*.

```
- (void)setFont:(NSFont *)font
```

Parameters

font

The font to use as the tab view label font.

Discussion
Tab height is adjusted automatically to accommodate a new font size. If the view allows truncating, tab labels are truncated as needed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- allowsTruncatedLabels (page 2685)
- font (page 2687)
- setAllowsTruncatedLabels: (page 2694)

Declared In
NSTabView.h

setTabViewType:

Sets the tab type to *tabViewType*.

```
- (void)setTabViewType:(NSTabViewType)tabViewType
```

Parameters

tabViewType

The tab type to display the tabs. The supported values are listed in `NSTabViewType` (page 2699)

Availability
Available in Mac OS X v10.0 and later.

See Also
- `tabViewType` (page 2698)

Declared In
`NSTabView.h`

tabViewItemAtIndex:

Returns the tab view item at *index* in the tab view's array of items.

```
- (NSTabViewItem *)tabViewItemAtIndex:(NSInteger)index
```

Parameters

index

The index at which to insert the tab view item. The *index* parameter is zero-based.

Return Value
The tab view item at the specified index.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `indexOfTabViewItem:` (page 2688)
- `insertTabViewItem:atIndex:` (page 2689)
- `tabViewItems` (page 2698)

Declared In
`NSTabView.h`

tabViewItemAtPoint:

Returns the tab view item at the specified point.

```
- (NSTabViewItem *)tabViewItemAtPoint:(NSPoint)point
```

Parameters

point

The hit point.

Return Value
The tab view item under the hit point, or `nil` if no tab view item is under that location.

Discussion
You can use this method to find a tab view item based on a user's mouse click.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTabView.h`

tabViewItems

Returns the receiver's array of tab view items.

```
- (NSArray *)tabViewItems
```

Return Value
An array of tab view items.

Discussion
A tab view keeps an array containing one tab view item for each tab in the view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `numberOfTabViewItems` (page 2690)
- `tabViewItemAtIndex:` (page 2697)

Declared In
`NSTabView.h`

tabViewType

Returns the tab type for the receiver.

```
- (NSTabViewType)tabViewType
```

Return Value
The tab type to display the tabs. The supported values are listed in `NSTabViewType` (page 2699)

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTabView.h`

takeSelectedTabViewItemFromSender:

Sets the selected tab view item to the selected item obtained from the sender.

```
- (void)takeSelectedTabViewItemFromSender:(id)sender
```

Parameters

sender

Typically the object that sent invoked the message.

Discussion

If *sender* responds to the `indexOfSelectedItem` method, this method invokes that method and selects the tab view item at the specified index.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTabView.h`

Constants

NSTabViewType

These constants specify the tab view's type as used by `tabViewType` (page 2698) and `setTabViewType:` (page 2697).

```
enum {
    NSTopTabsBezelBorder     = 0,
    NSLeftTabsBezelBorder    = 1,
    NSBottomTabsBezelBorder  = 2,
    NSRightTabsBezelBorder   = 3,
    NSNoTabsBezelBorder      = 4,
    NSNoTabsLineBorder       = 5,
    NSNoTabsNoBorder         = 6
};
typedef NSUInteger NSTabViewType;
```

Constants

`NSTopTabsBezelBorder`

The view includes tabs on the top of the view and has a bezeled border (the default).

Available in Mac OS X v10.0 and later.

Declared in `NSTabView.h`.

`NSNoTabsBezelBorder`

The view does not include tabs and has a bezeled border.

Available in Mac OS X v10.0 and later.

Declared in `NSTabView.h`.

`NSNoTabsLineBorder`

The view does not include tabs and has a lined border.

Available in Mac OS X v10.0 and later.

Declared in `NSTabView.h`.

`NSNoTabsNoBorder`
The view does not include tabs and has no border.

Available in Mac OS X v10.0 and later.

Declared in `NSTabView.h`.

`NSBottomTabsBezelBorder`
Tabs are on the bottom of the view with a bezeled border.

Available in Mac OS X v10.0 and later.

Declared in `NSTabView.h`.

`NSLeftTabsBezelBorder`
Tabs are on the left of the view with a bezeled border.

Available in Mac OS X v10.0 and later.

Declared in `NSTabView.h`.

`NSRightTabsBezelBorder`
Tabs are on the right of the view with a bezeled border.

Available in Mac OS X v10.0 and later.

Declared in `NSTabView.h`.

NSAppKitVersionNumberWithDirectionalTabs

This constant specifies the minimum version of the Application Kit that supports directional tabs.

```
#define NSAppKitVersionNumberWithDirectionalTabs 631.0
```

Constants

`NSAppKitVersionNumberWithDirectionalTabs`
The specific version of the AppKit framework that introduced support for directional tab items. Developers should not need to use this constant unless they are writing applications for Mac OS X v10.1 and earlier.

Available in Mac OS X v10.2 and later.

Declared in `NSTabView.h`.

NSTabViewItem Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTabViewItem.h
Companion guide	Tab Views
Related sample code	CoreRecipes From A View to A Movie From A View to A Picture iChatTheater TextSizingExample

Overview

An NSTabViewItem is a convenient way for presenting information in multiple pages. A tab view is usually distinguished by a row of tabs that give the visual appearance of folder tabs. When the user clicks a tab, the tab view displays a view page provided by your application. A tab view keeps a zero-based array of NSTabViewItems, one for each tab in the view.

Tasks

Creating a Tab View Item

`- initWithIdentifier:` (page 2704)
Performs default initialization for the receiver.

Working with Labels

`- drawLabel:inRect:` (page 2703)
Draws the receiver's label in *tabRect*, which is the area between the curved end caps.

- `label` (page 2705)
  Returns the label text for the receiver.
- `setLabel:` (page 2706)
  Sets the label text for the receiver to *`label`*.
- `sizeOfLabel:` (page 2707)
  Calculates the size of the receiver's label.

Checking the Tab Display State

- `tabState` (page 2708)
  Returns the current display state of the tab associated with the receiver.

Assigning an Identifier Object

- `identifier` (page 2704)
  Returns the receiver's optional identifier object.
- `setIdentifier:` (page 2705)
  Sets the receiver's optional identifier object to *`identifier`*.

Setting the Color

- `color` (page 2703)
  Returns the color for the receiver.
- `setColor:` (page 2705)
  Deprecated. NSTabViewItems use a color supplied by the current theme.

Assigning a View

- `view` (page 2709)
  Returns the view associated with the receiver.
- `setView:` (page 2707)
  Sets the view associated with the receiver to *`view`*.

Setting the Initial First Responder

- `initialFirstResponder` (page 2704)
  Returns the initial first responder for the view associated with the receiver.
- `setInitialFirstResponder:` (page 2706)
  Sets the initial first responder for the view associated with the receiver (the view that is displayed when a user clicks on the tab) to *`view`*.

Accessing the Parent Tab View

- `tabView` (page 2708)
  Returns the parent tab view for the receiver.

Getting and Setting Tooltips

- `toolTip` (page 2708)
  Returns the tooltip displayed for the tab view item
- `setToolTip:` (page 2706)
  Sets the tooltip displayed for the tab view item.

Instance Methods

color

Returns the color for the receiver.

```
- (NSColor *)color
```

Discussion
The color is specified by the current theme.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setColor:` (page 2705)

Declared In
`NSTabViewItem.h`

drawLabel:inRect:

Draws the receiver's label in *tabRect*, which is the area between the curved end caps.

```
- (void)drawLabel:(BOOL)shouldTruncateLabel inRect:(NSRect)tabRect
```

Discussion
If *shouldTruncateLabel* is `NO`, draws the full label in the rectangle specified by *tabRect*. If *shouldTruncateLabel* is `YES`, draws the truncated label. You can override this method to perform customized label drawing. For example, you might want to add an icon to each tab in the view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `sizeOfLabel:` (page 2707)

Declared In
NSTabViewItem.h

identifier

Returns the receiver's optional identifier object.

```
- (id)identifier
```

Discussion
To customize how your application works with tabs, you can initialize each tab view item with an identifier object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- initWithIdentifier: (page 2704)
- setIdentifier: (page 2705)

Declared In
NSTabViewItem.h

initialFirstResponder

Returns the initial first responder for the view associated with the receiver.

```
- (id)initialFirstResponder
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- setInitialFirstResponder: (page 2706)

Declared In
NSTabViewItem.h

initWithIdentifier:

Performs default initialization for the receiver.

```
- (id)initWithIdentifier:(id)identifier
```

Discussion
Sets the receiver's identifier object to *identifier*, if it is not nil. Use this method when creating tab view items programmatically.

Availability
Available in Mac OS X v10.0 and later.

See Also
- identifier (page 2704)
- setIdentifier: (page 2705)

Declared In
NSTabViewItem.h

label

Returns the label text for the receiver.

```
- (NSString *)label
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- setLabel: (page 2706)

Declared In
NSTabViewItem.h

setColor:

Deprecated. NSTabViewItems use a color supplied by the current theme.

```
- (void)setColor:(NSColor *)color
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- color (page 2703)

Declared In
NSTabViewItem.h

setIdentifier:

Sets the receiver's optional identifier object to *identifier*.

```
- (void)setIdentifier:(id)identifier
```

Discussion
To customize how your application works with tabs, you can specify an identifier object for each tab view item.

Availability
Available in Mac OS X v10.0 and later.

See Also
- identifier (page 2704)

- initWithIdentifier: (page 2704)

Declared In
NSTabViewItem.h

setInitialFirstResponder:

Sets the initial first responder for the view associated with the receiver (the view that is displayed when a user clicks on the tab) to *view*.

```
- (void)setInitialFirstResponder:(NSView *)view
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- initialFirstResponder (page 2704)

Related Sample Code
TextSizingExample

Declared In
NSTabViewItem.h

setLabel:

Sets the label text for the receiver to *label*.

```
- (void)setLabel:(NSString *)label
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- label (page 2705)

Related Sample Code
CoreRecipes

Declared In
NSTabViewItem.h

setToolTip:

Sets the tooltip displayed for the tab view item.

```
- (void)setToolTip:(NSString *)toolTip
```

Parameters
toolTip
A string representing the tooltip to be displayed.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `toolTip` (page 2708)

Declared In
`NSTabViewItem.h`

setView:

Sets the view associated with the receiver to *view*.

```
- (void)setView:(NSView *)view
```

Discussion
This is the view displayed when a user clicks the tab. When you set a new view, the old view is released.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `view` (page 2709)

Related Sample Code
CoreRecipes
TextSizingExample

Declared In
`NSTabViewItem.h`

sizeOfLabel:

Calculates the size of the receiver's label.

```
- (NSSize)sizeOfLabel:(BOOL)shouldTruncateLabel
```

Discussion
If *shouldTruncateLabel* is `NO`, returns the size of the receiver's full label. If *shouldTruncateLabel* is `YES`, returns the truncated size. If your application does anything to change the size of tab labels, such as overriding the `drawLabel:inRect:` (page 2703) method to add an icon to each tab, you should override `sizeOfLabel:` (page 2707) too so the NSTabView knows the correct size for the tab label.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawLabel:inRect:` (page 2703)
- `setFont:` (page 2696) (NSTabView)

Declared In
`NSTabViewItem.h`

tabState

Returns the current display state of the tab associated with the receiver.

```
- (NSTabState)tabState
```

Discussion
The possible values are `NSSelectedTab`, `NSBackgroundTab`, or `NSPressedTab`. Your application does not directly set the tab state.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTabViewItem.h`

tabView

Returns the parent tab view for the receiver.

```
- (NSTabView *)tabView
```

Discussion
Note that this is the tab view itself, not the view displayed when a user clicks the tab.

A tab view item normally learns about its parent tab view when it is inserted into the view's array of items. The NSTabView methods `addTabViewItem:` (page 2685) and `insertTabViewItem:atIndex:` (page 2689) set the tab view for the added or inserted item.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setView:` (page 2707)
- `view` (page 2709)

Declared In
`NSTabViewItem.h`

toolTip

Returns the tooltip displayed for the tab view item

```
- (NSString *)toolTip
```

Return Value
A string representing the tooltip to be displayed.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setToolTip:` (page 2706)

Declared In
NSTabViewItem.h

view

Returns the view associated with the receiver.

```
- (id)view
```

Discussion
This is the view displayed when a user clicks the tab.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setView: (page 2707)

Related Sample Code
iChatTheater
Reducer

Declared In
NSTabViewItem.h

Constants

NSTabState

These constants describe the current display state of a tab:

```
typedef enum _NSTabState {
    NSSelectedTab = 0,
    NSBackgroundTab = 1,
    NSPressedTab = 2
} NSTabState;
```

Constants

NSBackgroundTab
A tab that's not being displayed.
Available in Mac OS X v10.0 and later.
Declared in NSTabViewItem.h.

NSPressedTab
A tab that the user is in the process of clicking. That is, the user has pressed the mouse button while the cursor is over the tab but has not released the mouse button.
Available in Mac OS X v10.0 and later.
Declared in NSTabViewItem.h.

`NSSelectedTab`
: The tab that's being displayed.

  Available in Mac OS X v10.0 and later.

  Declared in `NSTabViewItem.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTabViewItem.h`

NSText Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSChangeSpelling NSIgnoreMisspelledWords NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSText.h
Companion guide	Text System Overview
Related sample code	DragNDropOutlineView EnhancedAudioBurn QTKitMovieShuffler SourceView STUCAuthoringDeviceCocoaSample

Class at a Glance

`NSText` declares the most general programmatic interface for objects that manage text. You usually use instances of its subclass, `NSTextView`.

Principal Attributes

- Draws text for user interface objects
- Uses a delegate
- Provides text editing capabilities
- Controls text attributes such as type size, font, and color

Commonly Used Methods

`readRTFDFromFile:` (page 2729)
Reads an `.rtf` or `.rtfd` file.

`writeRTFDToFile:atomically:` (page 2747)
Writes the receiver's text to a file.

`string` (page 2744)
Returns the receiver's text without attributes.

`RTFFromRange:` (page 2732)
Returns the receiver's text with attributes.

`RTFDFromRange:` (page 2731)
Returns the receiver's text with attributes and attachments.

Overview

`NSText` declares the most general programmatic interface for objects that manage text. You usually use instances of its subclass, `NSTextView`.

`NSTextView` extends the interface declared by `NSText` and provides much more sophisticated functionality than that declared in `NSText`.

`NSText` initialization creates an instance of a concrete subclass, such as `NSTextView`. Instances of any of these classes are generically called text objects.

Text objects are used by the Application Kit wherever text appears in interface objects: A text object draws the title of a window, the commands in a menu, the title of a button, and the items in a browser. Your application can also create text objects for its own purposes.

Adopted Protocols

NSChangeSpelling
`changeSpelling:` (page 3605)

NSIgnoreMisspelledWords
`ignoreSpelling:` (page 3692)

Tasks

Getting the Characters

`- string` (page 2744)
Returns the characters of the receiver's text.

Setting Graphics Attributes

- `setBackgroundColor:` (page 2734)
  Sets the receiver's background color to a given color.
- `backgroundColor` (page 2719)
  Returns the receiver's background color.
- `setDrawsBackground:` (page 2735)
  Controls whether the receiver draws its background.
- `drawsBackground` (page 2723)
  Returns a Boolean value that indicates whether the receiver draws its background.

Setting Behavioral Attributes

- `setEditable:` (page 2736)
  Controls whether the receiver allows the user to edit its text.
- `isEditable` (page 2724)
  Returns a Boolean value that indicates whether the receiver allows the user to edit text, `NO` if it doesn't.
- `setSelectable:` (page 2740)
  Controls whether the receiver allows the user to select its text.
- `isSelectable` (page 2726)
  Returns a Boolean value that indicates whether the receiver allows the user to select text, `NO` if it doesn't.
- `setFieldEditor:` (page 2736)
  Controls whether the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder.
- `isFieldEditor` (page 2725)
  Returns a Boolean value that indicates whether the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder.
- `setRichText:` (page 2739)
  Controls whether the receiver allows the user to apply attributes to specific ranges of the text.
- `isRichText` (page 2726)
  Returns a Boolean value that indicates whether the receiver allows the user to apply attributes to specific ranges of the text.
- `setImportsGraphics:` (page 2738)
  Controls whether the receiver allows the user to import files by dragging.
- `importsGraphics` (page 2724)
  Returns a Boolean value that indicates whether the receiver allows the user to import files by dragging.

Using the Font Panel and Menu

- `setUsesFontPanel:` (page 2742)
  Controls whether the receiver uses the Font panel and Font menu.
- `usesFontPanel` (page 2746)
  Returns a Boolean value that indicates whether the receiver uses the Font panel.

Using the Ruler

- `toggleRuler:` (page 2745)
  This action method shows or hides the ruler, if the receiver is enclosed in a scroll view.
- `isRulerVisible` (page 2726)
  Returns a Boolean value that indicates whether the receiver's enclosing scroll view shows its ruler.

Changing the Selection

- `setSelectedRange:` (page 2740)
  Selects the receiver's characters within *aRange*.
- `selectedRange` (page 2733)
  Returns the range of selected characters.

Replacing Text

- `replaceCharactersInRange:withRTF:` (page 2729)
  Replaces the characters in the given range with RTF text interpreted from the given RTF data.
- `replaceCharactersInRange:withRTFD:` (page 2730)
  Replaces the characters in the given range with RTFD text interpreted from the given RTFD data.
- `replaceCharactersInRange:withString:` (page 2731)
  Replaces the characters in the given range with those in the given string.
- `setString:` (page 2741)
  Replaces the receiver's entire text with *aString*, applying the formatting attributes of the old first character to its new contents.

Action Methods for Editing

- `selectAll:` (page 2732)
  This action method selects all of the receiver's text.
- `copy:` (page 2721)
  This action method copies the selected text onto the general pasteboard, in as many formats as the receiver supports.
- `cut:` (page 2722)
  This action method deletes the selected text and places it onto the general pasteboard, in as many formats as the receiver supports.
- `paste:` (page 2728)
  This action method pastes text from the general pasteboard at the insertion point or over the selection.
- `copyFont:` (page 2721)
  This action method copies the font information for the first character of the selection (or for the insertion point) onto the font pasteboard, as `NSFontPboardType`.
- `pasteFont:` (page 2728)
  This action method pastes font information from the font pasteboard onto the selected text or insertion point of a rich text object, or over all text of a plain text object.

- `copyRuler:` (page 2722)
  This action method copies the paragraph style information for first selected paragraph onto the ruler pasteboard, as `NSRulerPboardType`, and expands the selection to paragraph boundaries.
- `pasteRuler:` (page 2728)
  This action method pastes paragraph style information from the ruler pasteboard onto the selected paragraphs of a rich text object.
- `delete:` (page 2723)
  This action method deletes the selected text.

Changing the Font

- `changeFont:` (page 2720)
  This action method changes the font of the selection for a rich text object, or of all text for a plain text object.
- `setFont:` (page 2736)
  Sets the font of all the receiver's text to *aFont*.
- `setFont:range:` (page 2737)
  Sets the font of characters within *aRange* to *aFont*.
- `font` (page 2723)
  Returns the font of the first character in the receiver's text, or of the insertion point if there's no text.

Setting Text Alignment

- `alignCenter:` (page 2718)
  This action method applies center alignment to selected paragraphs (or all text if the receiver is a plain text object).
- `alignLeft:` (page 2718)
  This action method applies left alignment to selected paragraphs (or all text if the receiver is a plain text object).
- `alignRight:` (page 2719)
  This action method applies right alignment to selected paragraphs (or all text if the receiver is a plain text object).
- `alignment` (page 2718)
  Returns the alignment of the first paragraph (or all text if the receiver is a plain text object).
- `setAlignment:` (page 2733) Available in Mac OS X v10.0 through Mac OS X v10.5
  Sets the alignment of all the receiver's text to *mode*.

Setting Text Color

- `setTextColor:` (page 2741)
  Sets the text color of all characters in the receiver to *aColor*.
- `setTextColor:range:` (page 2742)
  Sets the text color of characters within *aRange* to *aColor*.

- `textColor` (page 2745)
  Returns the color of the receiver's first character, or for the insertion point if there's no text.

Writing Direction

- `baseWritingDirection` (page 2719)
  Returns the initial writing direction used to determine the actual writing direction for text.
- `setBaseWritingDirection:` (page 2734)
  Sets the initial writing direction used to determine the actual writing direction for text.

Setting Superscripting and Subscripting

- `superscript:` (page 2745)
  This action method applies a superscript attribute to selected text (or all text if the receiver is a plain text object), raising its baseline offset by a predefined amount.
- `subscript:` (page 2744)
  This action method applies a subscript attribute to selected text (or all text if the receiver is a plain text object), lowering its baseline offset by a predefined amount.
- `unscript:` (page 2746)
  This action method removes any superscripting or subscripting from selected text (or all text if the receiver is a plain text object).

Underlining Text

- `underline:` (page 2746)
  Adds the underline attribute to the selected text attributes if absent; removes the attribute if present.

Reading and Writing RTF Files

- `readRTFDFromFile:` (page 2729)
  Attempts to read the RTFD file at *`path`*, returning `YES` if successful and `NO` if not.
- `writeRTFDToFile:atomically:` (page 2747)
  Writes the receiver's text as RTF with attachments to a file or directory at *`path`*.
- `RTFDFromRange:` (page 2731)
  Returns an NSData object that contains an RTFD stream corresponding to the characters and attributes within *`aRange`*.
- `RTFFromRange:` (page 2732)
  Returns an NSData object that contains an RTF stream corresponding to the characters and attributes within *`aRange`*, omitting any attachment characters and attributes.

Checking Spelling

- `checkSpelling:` (page 2720)
  This action method searches for a misspelled word in the receiver's text.
- `showGuessPanel:` (page 2743)
  This action method opens the Spelling panel, allowing the user to make a correction during spell checking.

Constraining Size

- `setMaxSize:` (page 2738)
  Sets the receiver's maximum size to *aSize*.
- `maxSize` (page 2727)
  Returns the receiver's maximum size.
- `setMinSize:` (page 2739)
  Sets the receiver's minimum size to *aSize*.
- `minSize` (page 2727)
  Returns the receiver's minimum size.
- `setVerticallyResizable:` (page 2743)
  Controls whether the receiver changes its height to fit the height of its text.
- `isVerticallyResizable` (page 2727)
  Returns `YES` if the receiver automatically changes its height to accommodate the height of its text, `NO` if it doesn't.
- `setHorizontallyResizable:` (page 2737)
  Controls whether the receiver changes its width to fit the width of its text.
- `isHorizontallyResizable` (page 2725)
  Returns `YES` if the receiver automatically changes its width to accommodate the width of its text, `NO` if it doesn't.
- `sizeToFit` (page 2743) Deprecated in Mac OS X v10.2
  Resizes the receiver to fit its text.

Scrolling

- `scrollRangeToVisible:` (page 2732)
  Scrolls the receiver in its enclosing scroll view so the first characters of *aRange* are visible.

Setting the Delegate

- `setDelegate:` (page 2735)
  Sets the receiver's delegate.
- `delegate` (page 2722)
  Returns the receiver's delegate.

Instance Methods

alignCenter:

This action method applies center alignment to selected paragraphs (or all text if the receiver is a plain text object).

```
- (void)alignCenter:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alignLeft:` (page 2718)
- `alignRight:` (page 2719)
- `alignment` (page 2718)
- `setAlignment:` (page 2733)

Declared In
`NSText.h`

alignLeft:

This action method applies left alignment to selected paragraphs (or all text if the receiver is a plain text object).

```
- (void)alignLeft:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alignCenter:` (page 2718)
- `alignRight:` (page 2719)
- `alignment` (page 2718)
- `setAlignment:` (page 2733)

Declared In
`NSText.h`

alignment

Returns the alignment of the first paragraph (or all text if the receiver is a plain text object).

```
- (NSTextAlignment)alignment
```

Discussion
The returned value is one of the alignments described in "NSTextAlignment" (page 2747).

Text using `NSNaturalTextAlignment` is actually displayed using one of the other alignments, depending on the natural alignment of the text's script.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSText.h`

alignRight:

This action method applies right alignment to selected paragraphs (or all text if the receiver is a plain text object).

```
- (void)alignRight:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alignLeft:` (page 2718)
- `alignCenter:` (page 2718)
- `alignment` (page 2718)
- `setAlignment:` (page 2733)

Declared In
`NSText.h`

backgroundColor

Returns the receiver's background color.

```
- (NSColor *)backgroundColor
```

Return Value
The receiver's background color.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawsBackground` (page 2723)
- `setBackgroundColor:` (page 2734)

Declared In
`NSText.h`

baseWritingDirection

Returns the initial writing direction used to determine the actual writing direction for text.

```
- (NSWritingDirection)baseWritingDirection
```

Discussion
The Text system uses this value as a hint for calculating the actual direction for displaying Unicode characters. You should not need to call this method directly. If no writing direction is set, returns `NSWritingDirectionNatural`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setBaseWritingDirection:` (page 2734)

Declared In
`NSText.h`

changeFont:

This action method changes the font of the selection for a rich text object, or of all text for a plain text object.

```
- (void)changeFont:(id)sender
```

Discussion
If the receiver doesn't use the Font panel, this method does nothing.

This method changes the font by sending a `convertFont:` (page 1220) message to the shared NSFontManager and applying each NSFont returned to the appropriate text. See the `NSFontManager` class specification for more information on font conversion.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- usesFontPanel` (page 2746)

Declared In
`NSText.h`

checkSpelling:

This action method searches for a misspelled word in the receiver's text.

```
- (void)checkSpelling:(id)sender
```

Discussion
The search starts at the end of the selection and continues until it reaches a word suspected of being misspelled or the end of the text. If a word isn't recognized by the spelling server, a `showGuessPanel:` (page 2743) message then opens the Guess panel and allows the user to make a correction or add the word to the local dictionary.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `showGuessPanel:` (page 2743)

Declared In
`NSText.h`

copy:

This action method copies the selected text onto the general pasteboard, in as many formats as the receiver supports.

```
- (void)copy:(id)sender
```

Discussion
A plain text object uses `NSStringPboardType` for plain text, and a rich text object also uses `NSRTFPboardType`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `copyFont:` (page 2721)
- `copyRuler:` (page 2722)
- `cut:` (page 2722)
- `paste:` (page 2728)

Declared In
`NSText.h`

copyFont:

This action method copies the font information for the first character of the selection (or for the insertion point) onto the font pasteboard, as `NSFontPboardType`.

```
- (void)copyFont:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `copy:` (page 2721)
- `copyRuler:` (page 2722)
- `cut:` (page 2722)
- `paste:` (page 2728)

Declared In
`NSText.h`

copyRuler:

This action method copies the paragraph style information for first selected paragraph onto the ruler pasteboard, as `NSRulerPboardType`, and expands the selection to paragraph boundaries.

```
- (void)copyRuler:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `copy:` (page 2721)
- `copyFont:` (page 2721)
- `cut:` (page 2722)
- `paste:` (page 2728)

Declared In
`NSText.h`

cut:

This action method deletes the selected text and places it onto the general pasteboard, in as many formats as the receiver supports.

```
- (void)cut:(id)sender
```

Discussion
A plain text object uses `NSStringPboardType` for plain text, and a rich text object also uses `NSRTFPboardType`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `delete:` (page 2723)
- `copy:` (page 2721)
- `copyFont:` (page 2721)
- `copyRuler:` (page 2722)
- `paste:` (page 2728)

Declared In
`NSText.h`

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Return Value
The receiver's delegate, or `nil` if it has none.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDelegate:` (page 2735)

Declared In
`NSText.h`

delete:

This action method deletes the selected text.

```
- (void)delete:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cut:` (page 2722)

Declared In
`NSText.h`

drawsBackground

Returns a Boolean value that indicates whether the receiver draws its background.

```
- (BOOL)drawsBackground
```

Return Value
`YES` if the receiver draws its background, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `backgroundColor` (page 2719)
- `setDrawsBackground:` (page 2735)

Declared In
`NSText.h`

font

Returns the font of the first character in the receiver's text, or of the insertion point if there's no text.

```
- (NSFont *)font
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFont:` (page 2736)
- `setFont:range:` (page 2737)

Related Sample Code
Aperture Edit Plugin - Borders & Titles
TipWrapper
ToolbarSample

Declared In
`NSText.h`

importsGraphics

Returns a Boolean value that indicates whether the receiver allows the user to import files by dragging.

```
- (BOOL)importsGraphics
```

Return Value
`YES` if the receiver allows the user to import files by dragging, otherwise `NO`.

Discussion
A text object that accepts dragged files is also a rich text object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isRichText` (page 2726)
- `setImportsGraphics:` (page 2738)

Declared In
`NSText.h`

isEditable

Returns a Boolean value that indicates whether the receiver allows the user to edit text, `NO` if it doesn't.

```
- (BOOL)isEditable
```

Return Value
`YES` if the receiver allows the user to edit text, otherwise `NO`.

Discussion
You can change the receiver's text programmatically regardless of this setting.

If the receiver is editable, it's also selectable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isSelectable` (page 2726)

- setEditable: (page 2736)

Declared In
NSText.h

isFieldEditor

Returns a Boolean value that indicates whether the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder.

```
- (BOOL)isFieldEditor
```

Return Value
YES if the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder; NO if it accepts them as text input.

Discussion
See the NSWindow class specification for more information on field editors. By default, NSText objects don't behave as field editors.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setFieldEditor: (page 2736)

Declared In
NSText.h

isHorizontallyResizable

Returns YES if the receiver automatically changes its width to accommodate the width of its text, NO if it doesn't.

```
- (BOOL)isHorizontallyResizable
```

Discussion
By default, an NSText object is not horizontally resizable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isVerticallyResizable (page 2727)
- setHorizontallyResizable: (page 2737)

Related Sample Code
Sketch-112

Declared In
NSText.h

isRichText

Returns a Boolean value that indicates whether the receiver allows the user to apply attributes to specific ranges of the text.

```
- (BOOL)isRichText
```

Return Value
`YES` if the receiver allows the user to apply attributes to specific ranges of the text, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `importsGraphics` (page 2724)
- `setRichText:` (page 2739)

Declared In
`NSText.h`

isRulerVisible

Returns a Boolean value that indicates whether the receiver's enclosing scroll view shows its ruler.

```
- (BOOL)isRulerVisible
```

Return Value
`YES` if the receiver's enclosing scroll view shows its ruler, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `toggleRuler:` (page 2745)

Declared In
`NSText.h`

isSelectable

Returns a Boolean value that indicates whether the receiver allows the user to select text, `NO` if it doesn't.

```
- (BOOL)isSelectable
```

Return Value
`YES` if the receiver allows the user to select text, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isEditable` (page 2724)
- `setSelectable:` (page 2740)

Declared In
NSText.h

isVerticallyResizable

Returns YES if the receiver automatically changes its height to accommodate the height of its text, NO if it doesn't.

```
- (BOOL)isVerticallyResizable
```

Discussion
By default, an NSText object is vertically resizable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isHorizontallyResizable (page 2725)
- setVerticallyResizable: (page 2743)

Declared In
NSText.h

maxSize

Returns the receiver's maximum size.

```
- (NSSize)maxSize
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- minSize (page 2727)
- setMaxSize: (page 2738)

Declared In
NSText.h

minSize

Returns the receiver's minimum size.

```
- (NSSize)minSize
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- maxSize (page 2727)
- setMinSize: (page 2739)

Declared In
NSText.h

paste:

This action method pastes text from the general pasteboard at the insertion point or over the selection.

```
- (void)paste:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- copy: (page 2721)
- cut: (page 2722)
- pasteFont: (page 2728)
- pasteRuler: (page 2728)

Related Sample Code
GLUT

Declared In
NSText.h

pasteFont:

This action method pastes font information from the font pasteboard onto the selected text or insertion point of a rich text object, or over all text of a plain text object.

```
- (void)pasteFont:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- copyFont: (page 2721)
- pasteRuler: (page 2728)

Declared In
NSText.h

pasteRuler:

This action method pastes paragraph style information from the ruler pasteboard onto the selected paragraphs of a rich text object.

```
- (void)pasteRuler:(id)sender
```

Discussion
It doesn't apply to a plain text object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `copyFont:` (page 2721)
- `pasteRuler:` (page 2728)

Declared In
`NSText.h`

readRTFDFromFile:

Attempts to read the RTFD file at *path*, returning `YES` if successful and `NO` if not.

```
- (BOOL)readRTFDFromFile:(NSString *)path
```

Discussion
path should be the path for an `.rtf` file or an `.rtfd` file wrapper, not for the RTF file within an `.rtfd` file wrapper.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `writeRTFDToFile:atomically:` (page 2747)

Related Sample Code
GLSL Showpiece Lite
GLSLShowpiece

Declared In
`NSText.h`

replaceCharactersInRange:withRTF:

Replaces the characters in the given range with RTF text interpreted from the given RTF data.

```
- (void)replaceCharactersInRange:(NSRange)aRange withRTF:(NSData *)rtfData
```

Parameters

aRange
The range of characters to be replaced.

rtfData
The RTF data from which to derive the replacement string.

Discussion
This method applies only to rich text objects.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 2950) or `shouldChangeTextInRange:replacementString:` (page 2949) to include this method in an undoable action.

This method is designed for transferring text from out-of-process sources such as the pasteboard. In most cases, programmatic modification of the text is best done by operating on the text storage directly, using the general methods of `NSMutableAttributedString`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `replaceCharactersInRange:withRTFD:` (page 2730)
- `replaceCharactersInRange:withString:` (page 2731)

Related Sample Code
CocoaSpeechSynthesisExample

Declared In
`NSText.h`

replaceCharactersInRange:withRTFD:

Replaces the characters in the given range with RTFD text interpreted from the given RTFD data.

```
- (void)replaceCharactersInRange:(NSRange)aRange withRTFD:(NSData *)rtfdData
```

Parameters

aRange
The range of characters to be replaced.

rtfdData
The RTFD data from which to derive the replacement string.

Discussion
This method applies only to rich text objects.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 2950) or `shouldChangeTextInRange:replacementString:` (page 2949) to include this method in an undoable action.

This method is designed for transferring text from out-of-process sources such as the pasteboard. In most cases, programmatic modification of the text is best done by operating on the text storage directly, using the general methods of `NSMutableAttributedString`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `replaceCharactersInRange:withRTF:` (page 2729)
- `replaceCharactersInRange:withString:` (page 2731)

Declared In
`NSText.h`

replaceCharactersInRange:withString:

Replaces the characters in the given range with those in the given string.

```
- (void)replaceCharactersInRange:(NSRange)aRange withString:(NSString *)aString
```

Parameters

aRange

The range of characters to be replaced.

aString

The replacement string.

Discussion

For a rich text object, the text of *aString* is assigned the formatting attributes of the first character of the text it replaces, or of the character immediately before *aRange* if the range's length is 0. If the range's location is 0, the formatting attributes of the first character in the receiver are used.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 2950) or `shouldChangeTextInRange:replacementString:` (page 2949) to include this method in an undoable action.

In most cases, programmatic modification of the text is best done by operating on the text storage directly, using the general methods of `NSMutableAttributedString`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `replaceCharactersInRange:withRTF:` (page 2729)
- `replaceCharactersInRange:withRTFD:` (page 2730)

Related Sample Code

CocoaSpeechSynthesisExample

STUCAuthoringDeviceCocoaSample

Declared In

`NSText.h`

RTFDFromRange:

Returns an NSData object that contains an RTFD stream corresponding to the characters and attributes within *aRange*.

```
- (NSData *)RTFDFromRange:(NSRange)aRange
```

Discussion

Raises an `NSRangeException` if any part of *aRange* lies beyond the end of the receiver's characters.

When writing data to the pasteboard, you can use the NSData object as the first argument to `NSPasteboard`'s `setData:forType:` (page 1898) method, with a second argument of `NSRTFDPboardType`.

Availability

Available in Mac OS X v10.0 and later.

See Also
- RTFFromRange: (page 2732)

Declared In
NSText.h

RTFFromRange:

Returns an NSData object that contains an RTF stream corresponding to the characters and attributes within *aRange*, omitting any attachment characters and attributes.

```
- (NSData *)RTFFromRange:(NSRange)aRange
```

Discussion
Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

When writing data to the pasteboard, you can use the NSData object as the first argument to NSPasteboard's setData:forType: (page 1898) method, with a second argument of NSRTFPboardType.

Availability
Available in Mac OS X v10.0 and later.

See Also
- RTFDFromRange: (page 2731)

Declared In
NSText.h

scrollRangeToVisible:

Scrolls the receiver in its enclosing scroll view so the first characters of *aRange* are visible.

```
- (void)scrollRangeToVisible:(NSRange)aRange
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BackgroundExporter
CocoaSpeechSynthesisExample
Quartz Composer WWDC 2005 TextEdit
STUCAuthoringDeviceCocoaSample
TextSizingExample

Declared In
NSText.h

selectAll:

This action method selects all of the receiver's text.

```
- (void)selectAll:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Aperture Edit Plugin - Borders & Titles
CIAnnotation
GLUT

Declared In
NSText.h

selectedRange

Returns the range of selected characters.

```
- (NSRange)selectedRange
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- setSelectedRange: (page 2740)

Related Sample Code
CocoaSpeechSynthesisExample
Quartz Composer WWDC 2005 TextEdit
STUCAuthoringDeviceCocoaSample
TextLinks

Declared In
NSText.h

setAlignment:

Sets the alignment of all the receiver's text to *mode*.

```
- (void)setAlignment:(NSTextAlignment)mode
```

Discussion
The value of *mode* must be one of the alignments described in "NSTextAlignment" (page 2747).

Text using NSNaturalTextAlignment is actually displayed using one of the other alignments, depending on the natural alignment of the text's script.

This method does not include undo support by default. Clients must invoke shouldChangeTextInRanges:replacementStrings: (page 2950) or shouldChangeTextInRange:replacementString: (page 2949) to include this method in an undoable action.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alignment` (page 2718)
- `alignLeft:` (page 2718)
- `alignCenter:` (page 2718)
- `alignRight:` (page 2719)

Related Sample Code
OpenCL NBody Simulation Example
TextSizingExample

Declared In
`NSText.h`

setBackgroundColor:

Sets the receiver's background color to a given color.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Parameters

aColor
The background color for the receiver.

Discussion
This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 2950) or `shouldChangeTextInRange:replacementString:` (page 2949) to include this method in an undoable action.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDrawsBackground:` (page 2735)
- `backgroundColor` (page 2719)

Declared In
`NSText.h`

setBaseWritingDirection:

Sets the initial writing direction used to determine the actual writing direction for text.

```
- (void)setBaseWritingDirection:(NSWritingDirection)writingDirection
```

Discussion
If you know the base writing direction of the text you are rendering, you can use this method to specify that direction to the text system.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 2950) or `shouldChangeTextInRange:replacementString:` (page 2949) to include this method in an undoable action.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `baseWritingDirection` (page 2719)

Declared In
`NSText.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)anObject
```

Parameters

anObject
The delegate for the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `delegate` (page 2722)

Declared In
`NSText.h`

setDrawsBackground:

Controls whether the receiver draws its background.

```
- (void)setDrawsBackground:(BOOL)flag
```

Parameters

flag
If *flag* is `YES`, the receiver fills its background with the background color, if *flag* is `NO`, it doesn't.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBackgroundColor:` (page 2734)
- `drawsBackground` (page 2723)

Declared In
`NSText.h`

setEditable:

Controls whether the receiver allows the user to edit its text.

```
- (void)setEditable:(BOOL)flag
```

Parameters

flag

If *flag* is YES, the receiver allows the user to edit text and attributes; if *flag* is NO, it doesn't.

Discussion

You can change the receiver's text programmatically regardless of this setting. If the receiver is made editable, it's also made selectable. NSText objects are by default editable.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setSelectable: (page 2740)
- isEditable (page 2724)

Declared In

NSText.h

setFieldEditor:

Controls whether the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder.

```
- (void)setFieldEditor:(BOOL)flag
```

Parameters

flag

If *flag* is YES, the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder; if *flag* is NO, it doesn't, instead accepting these characters as text input.

Discussion

See the NSWindow class specification for more information on field editors. By default, NSText objects don't behave as field editors.

Availability

Available in Mac OS X v10.0 and later.

See Also

- isFieldEditor (page 2725)

Declared In

NSText.h

setFont:

Sets the font of all the receiver's text to *aFont*.

```
- (void)setFont:(NSFont *)aFont
```

Discussion
This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 2950) or `shouldChangeTextInRange:replacementString:` (page 2949) to include this method in an undoable action.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFont:range:` (page 2737)
- `font` (page 2723)

Related Sample Code
FunHouse
OpenCL NBody Simulation Example
TipWrapper
ToolbarSample
UIElementInspector

Declared In
`NSText.h`

setFont:range:

Sets the font of characters within *aRange* to *aFont*.

```
- (void)setFont:(NSFont *)aFont range:(NSRange)aRange
```

Discussion
This method applies only to a rich text object.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 2950) or `shouldChangeTextInRange:replacementString:` (page 2949) to include this method in an undoable action.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFont:` (page 2736)
- `font` (page 2723)

Declared In
`NSText.h`

setHorizontallyResizable:

Controls whether the receiver changes its width to fit the width of its text.

```
- (void)setHorizontallyResizable:(BOOL)flag
```

Discussion
If *flag* is YES it does; if *flag* is NO it doesn't.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setVerticallyResizable: (page 2743)
- isHorizontallyResizable (page 2725)

Related Sample Code
FunHouse
GLUT
Quartz Composer WWDC 2005 TextEdit
Sketch-112
TextSizingExample

Declared In
NSText.h

setImportsGraphics:

Controls whether the receiver allows the user to import files by dragging.

```
- (void)setImportsGraphics:(BOOL)flag
```

Parameters
flag
If *flag* is YES, the receiver allows the user to import files by dragging; if *flag* is NO, it doesn't.

Discussion
If the receiver is set to accept dragged files, it's also made a rich text object. Subclasses may or may not accept dragged files by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setRichText: (page 2739)
- importsGraphics (page 2724)

Declared In
NSText.h

setMaxSize:

Sets the receiver's maximum size to *aSize*.

```
- (void)setMaxSize:(NSSize)aSize
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMinSize:` (page 2739)
- `maxSize` (page 2727)

Related Sample Code
FunHouse
Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility
Sketch-112
TextSizingExample

Declared In
`NSText.h`

setMinSize:

Sets the receiver's minimum size to *aSize*.

```
- (void)setMinSize:(NSSize)aSize
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMaxSize:` (page 2738)
- `minSize` (page 2727)

Related Sample Code
FunHouse
Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility
Sketch-112
TextSizingExample

Declared In
`NSText.h`

setRichText:

Controls whether the receiver allows the user to apply attributes to specific ranges of the text.

```
- (void)setRichText:(BOOL)flag
```

Parameters

flag

If *flag* is `YES` the receiver allows the user to apply attributes to specific ranges of the text; if *flag* is `NO` it doesn't.

Discussion
If *flag* is NO, the receiver is also set not to accept dragged files. Subclasses may or may not let the user apply multiple attributes to the text and accept drag files by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isRichText (page 2726)
- setImportsGraphics: (page 2738)

Declared In
NSText.h

setSelectable:

Controls whether the receiver allows the user to select its text.

```
- (void)setSelectable:(BOOL)flag
```

Parameters
flag
If *flag* is YES, the receiver allows the user to select text; if *flag* is NO, it doesn't.

Discussion
You can set selections programmatically regardless of this setting. If the receiver is made not selectable, it's also made not editable. NSText objects are by default editable and selectable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setEditable: (page 2736)
- isSelectable (page 2726)

Declared In
NSText.h

setSelectedRange:

Selects the receiver's characters within *aRange*.

```
- (void)setSelectedRange:(NSRange)aRange
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectedRange (page 2733)

Declared In
NSText.h

setString:

Replaces the receiver's entire text with *aString*, applying the formatting attributes of the old first character to its new contents.

```
- (void)setString:(NSString *)aString
```

Discussion
This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 2950) or `shouldChangeTextInRange:replacementString:` (page 2949) to include this method in an undoable action.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CoreRecipes
DeviceListener
TextSizingExample
TipWrapper
XMLBrowser

Declared In
`NSText.h`

setTextColor:

Sets the text color of all characters in the receiver to *aColor*.

```
- (void)setTextColor:(NSColor *)aColor
```

Discussion
Removes the text color attribute if *aColor* is `nil`.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 2950) or `shouldChangeTextInRange:replacementString:` (page 2949) to include this method in an undoable action.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTextColor:range:` (page 2742)
- `textColor` (page 2745)

Related Sample Code
FunHouse
TextSizingExample
ToolbarSample
UIElementInspector

Declared In
NSText.h

setTextColor:range:

Sets the text color of characters within *aRange* to *aColor*.

```
- (void)setTextColor:(NSColor *)aColor range:(NSRange)aRange
```

Discussion
Removes the text color attribute if *aColor* is nil. This method applies only to rich text objects.

This method does not include undo support by default. Clients must invoke shouldChangeTextInRanges:replacementStrings: (page 2950) or shouldChangeTextInRange:replacementString: (page 2949) to include this method in an undoable action.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTextColor: (page 2741)
- textColor (page 2745)

Related Sample Code
TextViewDelegate

Declared In
NSText.h

setUsesFontPanel:

Controls whether the receiver uses the Font panel and Font menu.

```
- (void)setUsesFontPanel:(BOOL)flag
```

Parameters

flag

If *flag* is YES, the receiver responds to messages from the Font panel and from the Font menu and updates the Font panel with the selection font whenever it changes. If *flag* is NO the receiver doesn't do any of these actions.

Discussion
By default, an NSText object uses the Font panel and menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- usesFontPanel (page 2746)

Declared In
NSText.h

setVerticallyResizable:

Controls whether the receiver changes its height to fit the height of its text.

```
- (void)setVerticallyResizable:(BOOL)flag
```

Discussion
If *flag* is `YES` it does; if *flag* is `NO` it doesn't.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHorizontallyResizable:` (page 2737)
- `isVerticallyResizable` (page 2727)

Related Sample Code
CIAnnotation
Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility
Sketch-112
TextSizingExample

Declared In
`NSText.h`

showGuessPanel:

This action method opens the Spelling panel, allowing the user to make a correction during spell checking.

```
- (void)showGuessPanel:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `checkSpelling:` (page 2720)

Declared In
`NSText.h`

sizeToFit

Resizes the receiver to fit its text.

```
- (void)sizeToFit
```

Discussion
The text view will not be sized any smaller than its minimum size, however.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isHorizontallyResizable` (page 2725)
- `isVerticallyResizable` (page 2727)

Related Sample Code
CIAnnotation
TextSizingExample
TipWrapper

Declared In
`NSText.h`

string

Returns the characters of the receiver's text.

```
- (NSString *)string
```

Return Value
The characters of the receiver's text.

Discussion
For performance reasons, this method returns the current backing store of the text object. If you want to maintain a snapshot of this as you manipulate the text storage, you should make a copy of the appropriate substring.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setString:` (page 2741)

Related Sample Code
CocoaSpeechSynthesisExample
FinalCutPro_AppleEvents
FunHouse
JSInterpreter
SimpleComboBox

Declared In
`NSText.h`

subscript:

This action method applies a subscript attribute to selected text (or all text if the receiver is a plain text object), lowering its baseline offset by a predefined amount.

```
- (void)subscript:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `subscript:` (page 2744)
- `unscript:` (page 2746)

`lowerBaseline:` (page 2906) (`NSTextView`)

Declared In
`NSText.h`

superscript:

This action method applies a superscript attribute to selected text (or all text if the receiver is a plain text object), raising its baseline offset by a predefined amount.

```
- (void)superscript:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `subscript:` (page 2744)
- `unscript:` (page 2746)

`raiseBaseline:` (page 2911) (`NSTextView`)

Declared In
`NSText.h`

textColor

Returns the color of the receiver's first character, or for the insertion point if there's no text.

```
- (NSColor *)textColor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTextColor:` (page 2741)
- `setTextColor:range:` (page 2742)

Declared In
`NSText.h`

toggleRuler:

This action method shows or hides the ruler, if the receiver is enclosed in a scroll view.

```
- (void)toggleRuler:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSText.h

underline:

Adds the underline attribute to the selected text attributes if absent; removes the attribute if present.

```
- (void)underline:(id)sender
```

Discussion
If there is a selection and the first character of the selected range has any form of underline on it, or if there is no selection and the typing attributes have any form of underline, then underline is removed; otherwise a single simple underline is added.

Operates on the selected range if the receiver contains rich text. For plain text the range is the entire contents of the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSText.h

unscript:

This action method removes any superscripting or subscripting from selected text (or all text if the receiver is a plain text object).

```
- (void)unscript:(id)sender
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- subscript: (page 2744)
- superscript: (page 2745)
raiseBaseline: (page 2911) (NSTextView)
lowerBaseline: (page 2906) (NSTextView)

Declared In
NSText.h

usesFontPanel

Returns a Boolean value that indicates whether the receiver uses the Font panel.

```
- (BOOL)usesFontPanel
```

Return Value
YES if the receiver uses the Font panel, otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setUsesFontPanel:` (page 2742)

Declared In
`NSText.h`

writeRTFDToFile:atomically:

Writes the receiver's text as RTF with attachments to a file or directory at *path*.

```
- (BOOL)writeRTFDToFile:(NSString *)path atomically:(BOOL)atomicFlag
```

Discussion
Returns `YES` on success and `NO` on failure. If *atomicFlag* is `YES`, attempts to write the file safely so that an existing file at *path* is not overwritten, nor does a new file at *path* actually get created, unless the write is successful.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `RTFFromRange:` (page 2732)
- `RTFDFromRange:` (page 2731)
- `readRTFDFromFile:` (page 2729)

Declared In
`NSText.h`

Constants

NSTextAlignment

These constants specify text alignment.

```
typedef enum _NSTextAlignment {
    NSLeftTextAlignment      = 0,
    NSRightTextAlignment     = 1,
    NSCenterTextAlignment    = 2,
    NSJustifiedTextAlignment = 3,
    NSNaturalTextAlignment   = 4
} NSTextAlignment;
```

Constants
`NSLeftTextAlignment`
Text is visually left aligned.
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSRightTextAlignment`
Text is visually right aligned.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSCenterTextAlignment`
Text is visually center aligned.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSJustifiedTextAlignment`
Text is justified.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSNaturalTextAlignment`
Use the natural alignment of the text's script.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

Declared In
`NSText.h`

NSWritingDirection

These constants specify the writing directions:

```
enum {
    NSWritingDirectionNatural     = -1,
    NSWritingDirectionLeftToRight = 0,
    NSWritingDirectionRightToLeft
};
typedef NSInteger NSWritingDirection;
```

Constants
`NSWritingDirectionNatural`
The writing direction is determined using the Unicode Bidi Algorithm rules P2 and P3. Default.

Available in Mac OS X v10.4 and later.

Declared in `NSText.h`.

`NSWritingDirectionLeftToRight`
The writing direction is left to right.

Available in Mac OS X v10.2 and later.

Declared in `NSText.h`.

`NSWritingDirectionRightToLeft`
The writing direction is right to left.

Available in Mac OS X v10.2 and later.

Declared in `NSText.h`.

Additional Writing Directions

Additional values to be added to `NSWritingDirectionLeftToRight` (page 2748) or `NSWritingDirectionRightToLeft` (page 2748), when used with `NSWritingDirectionAttributeName` (page 274).

```
enum {
    NSTextWritingDirectionEmbedding     = (0 << 1),
    NSTextWritingDirectionOverride      = (1 << 1)
};
```

Constants

`NSTextWritingDirectionEmbedding`

Direction is embedded.

Available in Mac OS X v10.6 and later.

Declared in `NSText.h`.

`NSTextWritingDirectionOverride`

Direction override

Available in Mac OS X v10.6 and later.

Declared in `NSText.h`.

Movement Codes

These constants specify the reason for a change of editing focus among text fields, in essence answering the question “why am I leaving the field?”

```
enum {
   NSIllegalTextMovement = 0,
   NSReturnTextMovement  = 0x10,
   NSTabTextMovement     = 0x11,
   NSBacktabTextMovement = 0x12,
   NSLeftTextMovement    = 0x13,
   NSRightTextMovement   = 0x14,
   NSUpTextMovement      = 0x15,
   NSDownTextMovement    = 0x16,
   NSCancelTextMovement  = 0x17,
  NSOtherTextMovement    = 0
};
```

Constants

`NSIllegalTextMovement`

Currently unused.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSReturnTextMovement`

The Return key was pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSTabTextMovement`
The Tab key was pressed.
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSBacktabTextMovement`
The Backtab (Shift-Tab) key was pressed.
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSLeftTextMovement`
The left arrow key was pressed.
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSRightTextMovement`
The right arrow key was pressed.
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSUpTextMovement`
The up arrow key was pressed.
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSDownTextMovement`
The down arrow key was pressed.
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSCancelTextMovement`
The user cancelled the completion.
Available in Mac OS X v10.3 and later.
Declared in `NSText.h`.

`NSOtherTextMovement`
The user performed some undefined action.
Available in Mac OS X v10.3 and later.
Declared in `NSText.h`.

Discussion
They are the possible values for the `NSTextMovement` key of the `NSTextDidEndEditingNotification` (page 2752)`userInfo` dictionary. The field editor makes sure that these are the values sent when the user presses the Tab, Backtab, or Return key while editing. The control then uses this information to decide where to send focus next.

Declared In
`NSText.h`

Commonly-used Unicode characters

These constants specify several commonly used Unicode characters.

```
enum {
   NSParagraphSeparatorCharacter = 0x2029,
   NSLineSeparatorCharacter      = 0x2028,
   NSTabCharacter                = 0x0009,
   NSFormFeedCharacter           = 0x000c,
   NSNewlineCharacter            = 0x000a,
   NSCarriageReturnCharacter     = 0x000d,
   NSEnterCharacter              = 0x0003,
   NSBackspaceCharacter          = 0x0008,
   NSBackTabCharacter            = 0x0019,
   NSDeleteCharacter             = 0x007f
};
```

Constants

`NSParagraphSeparatorCharacter`
The paragraph separator character: `0x2029`
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSLineSeparatorCharacter`
The line separator character: `0x2028`
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSTabCharacter`
The tab character: `0x0009`
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSBackTabCharacter`
The back tab character: `0x0019`
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSFormFeedCharacter`
The form feed character: `0x000c`
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSNewlineCharacter`
The newline character: `0x000a`
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSCarriageReturnCharacter`
The carriage return character: `0x000d`
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSEnterCharacter`
The enter character: `0x0003`
Available in Mac OS X v10.0 and later.
Declared in `NSText.h`.

`NSBackspaceCharacter`
The backspace character: `0x0008`

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSDeleteCharacter`
The delete character: `0x007f`

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

Declared In
`NSText.h`

Notifications

NSTextDidBeginEditingNotification

Posted when an `NSText` object begins any operation that changes characters or formatting attributes.

The notification object is the notifying `NSText` object. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSText.h`

NSTextDidChangeNotification

Posted after an `NSText` object performs any operation that changes characters or formatting attributes.

The notification object is the notifying `NSText` object. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSText.h`

NSTextDidEndEditingNotification

Posted when focus leaves an `NSText` object, whether or not any operation has changed characters or formatting attributes.

The notification object is the notifying `NSText` object. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSTextMovement"`	Possible movement code values are described in "Movement Codes" (page 2749).

Note: It is common for `NSTextDidEndEditingNotification` (page 2752) to be sent without a matching `NSTextDidBeginEditingNotification` (page 2752). The begin notification is only sent if the user actually makes changes (that is, types something or changes formatting attributes). However, the end notification is sent when focus leaves the text view, regardless of whether there was a change.

This distinction enables an application to know whether the user actually made a change to the text or just clicked in the text view and then clicked outside it. In both cases, `NSTextDidEndEditingNotification` (page 2752) is sent, but to tell the difference, the application can listen for `NSTextDidBeginEditingNotification` (page 2752).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSText.h`

NSTextAttachment Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTextAttachment.h
Companion guides	Text System Overview Text Attachment Programming Topics
Related sample code	CoreRecipes GLUT Quartz Composer WWDC 2005 TextEdit

Overview

`NSTextAttachment` objects are used by the `NSAttributedString` class cluster as the values for attachment attributes (stored in the attributed string under the key named `NSAttachmentAttributeName`). The objects you create with this class are referred to as text attachment objects, or when no confusion will result, as text attachments or merely attachments.

A text attachment object contains an NSFileWrapper object, which in turn holds the contents of the attached file. It also uses a cell object conforming to the NSTextAttachmentCell protocol to draw and handle mouse events. Most of the behavior of a text attachment is relegated to the file wrapper and the attachment cell. See the corresponding class and protocol specifications for more information.

See the NSAttributedString and NSTextView class specifications for general information on text attachments.

Adopted Protocols

NSCoding

- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Initializing an NSTextAttachment Object

- `initWithFileWrapper:` (page 2757)
  Initializes a newly allocated `NSTextAttachment` object to contain the given file wrapper.

Setting the File Wrapper

- `setFileWrapper:` (page 2758)
  Sets the receiver's file wrapper.
- `fileWrapper` (page 2757)
  Returns the receiver's file wrapper.

Setting the Attachment Cell

- `setAttachmentCell:` (page 2758)
  Sets the object used to draw the icon for the receiver and to handle mouse events.
- `attachmentCell` (page 2756)
  Returns the object used to draw the icon for the receiver and to handle mouse events.

Instance Methods

attachmentCell

Returns the object used to draw the icon for the receiver and to handle mouse events.

```
- (id < NSTextAttachmentCell >)attachmentCell
```

Return Value
The object used to draw the icon for the receiver and to handle mouse events.

Discussion
An `NSTextAttachment` object by default uses an `NSTextAttachmentCell` object that displays the attached file's icon, or its contents if the file contains an image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `fileWrapper` (page 2757)
- `image` (page 562) (`NSCell`)
- `icon` (`NSFileWrapper`)
- `setAttachmentCell:` (page 2758)

Declared In
NSTextAttachment.h

fileWrapper

Returns the receiver's file wrapper.

```
- (NSFileWrapper *)fileWrapper
```

Return Value
The receiver's file wrapper.

Discussion
The file wrapper holds the contents of the attached file.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setFileWrapper: (page 2758)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSTextAttachment.h

initWithFileWrapper:

Initializes a newly allocated NSTextAttachment object to contain the given file wrapper.

```
- (id)initWithFileWrapper:(NSFileWrapper *)aWrapper
```

Parameters
aWrapper
The file wrapper for the receiver.

Return Value
The receiver initialized to contain *aWrapper* and use an NSTextAttachmentCell as its attachment cell.

Discussion
This method is the designated initializer for the NSTextAttachment class.

If *aWrapper* contains an image file that the receiver can interpret as an NSImage object, sets the attachment cell's image to the NSImage rather than to the icon of *aWrapper*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setFileWrapper: (page 2758)
- setAttachmentCell: (page 2758)

Related Sample Code
CoreRecipes

Declared In
NSTextAttachment.h

setAttachmentCell:

Sets the object used to draw the icon for the receiver and to handle mouse events.

```
- (void)setAttachmentCell:(id < NSTextAttachmentCell >)aCell
```

Parameters

aCell

The object used to draw the icon for the receiver and to handle mouse events.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setFileWrapper: (page 2758)
- setImage: (page 589) (NSCell)
- icon (NSFileWrapper)
- attachmentCell (page 2756)

Declared In
NSTextAttachment.h

setFileWrapper:

Sets the receiver's file wrapper.

```
- (void)setFileWrapper:(NSFileWrapper *)aWrapper
```

Parameters

aWrapper

The file wrapper for the receiver.

Discussion
The file wrapper holds the contents of the attached file.

Availability
Available in Mac OS X v10.0 and later.

See Also
- fileWrapper (page 2757)

Declared In
NSTextAttachment.h

Constants

Attachment Character

This character is used to denote an attachment.

```
enum {
   NSAttachmentCharacter = 0xfffc
};
```

Constants

`NSAttachmentCharacter`

Specifies a character that denotes attachment an attachment.

Available in Mac OS X v10.0 and later.

Declared in `NSTextAttachment.h`.

Declared In

`NSTextAttachment.h`

NSTextAttachmentCell Class Reference

Inherits from	NSCell : NSObject
Conforms to	NSTextAttachmentCell NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSTextAttachment.h
Companion guides	Text System Overview Text Attachment Programming Topics
Related sample code	Quartz Composer WWDC 2005 TextEdit

Overview

`NSTextAttachmentCell` implements the functionality of the `NSTextAttachmentCell` protocol. See the NSTextAttachmentCell protocol specification for a general discussion of the protocol's methods. This specification describes only those methods whose implementations have features peculiar to this class.

Adopted Protocols

NSTextAttachmentCell

- `- attachment` (page 3848)
- `- cellBaselineOffset` (page 3849)
- `- cellFrameForTextContainer:proposedLineFragment:glyphPosition:characterIndex:` (page 3849)
- `- cellSize` (page 3849)
- `- drawWithFrame:inView:` (page 3850)
- `- drawWithFrame:inView:characterIndex:` (page 3850)
- `- drawWithFrame:inView:characterIndex:layoutManager:` (page 3850)
- `- highlight:withFrame:inView:` (page 3851)
- `- setAttachment:` (page 3851)
- `- trackMouse:inRect:ofView:untilMouseUp:` (page 3852)

- `trackMouse:inRect:ofView:atCharacterIndex:untilMouseUp:` (page 3851)
- `wantsToTrackMouse` (page 3853)
- `wantsToTrackMouseForEvent:inRect:ofView:atCharacterIndex:` (page 3853)

NSTextBlock Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSTextTable.h
Availability	Available in Mac OS X v10.4 and later.
Companion guides	Text System Overview Text Layout Programming Guide

Overview

`NSTextBlock` objects represent a block of text laid out in a subregion of the text container. Text blocks appear as attributes on paragraphs, as part of the paragraph style.

The most important subclass is `NSTextTableBlock`, which represents a block of text that appears as a cell in a table. The table itself is represented by a separate class, `NSTextTable`, which is referenced by all of its `NSTextTableBlock` objects and which controls their sizing and positioning.

Tasks

Creating Text Blocks

`- init` (page 2768)
Initializes and returns an empty text block object.

Working with Dimensions of Content

`- setValue:type:forDimension:` (page 2770)
Sets a dimension of the text block.

`- valueForDimension:` (page 2772)
Returns the value of the specified text block dimension.

- `valueTypeForDimension:` (page 2773)
  Returns the value type of the specified text block dimension.
- `setContentWidth:type:` (page 2770)
  Sets the width of the text block.
- `contentWidth` (page 2766)
  Returns the width of the text block.
- `contentWidthValueType` (page 2767)
  Returns the type of value stored for the text block width.

Getting and Setting Margins, Borders, and Padding

- `setWidth:type:forLayer:edge:` (page 2772)
  Sets the width of a specified edge of a specified layer of the text block.
- `setWidth:type:forLayer:` (page 2771)
  Sets the width of all edges of a specified layer of the text block.
- `widthForLayer:edge:` (page 2774)
  Returns the width of an edge of a specified layer of the text block.
- `widthValueTypeForLayer:edge:` (page 2774)
  Returns the value type of an edge of a specified layer of the text block.

Getting and Setting Alignment

- `setVerticalAlignment:` (page 2771)
  Sets the vertical alignment of the text block.
- `verticalAlignment` (page 2773)
  Returns the vertical alignment of the text block.

Working with Color

- `setBackgroundColor:` (page 2769)
  Sets the background color of the text block.
- `backgroundColor` (page 2765)
  Returns the background color of the text block.
- `setBorderColor:forEdge:` (page 2769)
  Sets the border color of the specified edge of the text block.
- `setBorderColor:` (page 2769)
  Sets the color of all borders of the text block.
- `borderColorForEdge:` (page 2765)
  Returns the border color of the specified text block edge.

Determining Size and Position of a Text Block

- `rectForLayoutAtPoint:inRect:textContainer:characterRange:` (page 2768)
  Returns the rectangle within which glyphs should be laid out for the specified arguments.
- `boundsRectForContentRect:inRect:textContainer:characterRange:` (page 2766)
  Returns the rectangle the text in the block actually occupies, including padding, borders, and margins.

Drawing Colors and Decorations

- `drawBackgroundWithFrame:inView:characterRange:layoutManager:` (page 2767)
  Called by the layout manager to draw any colors and other decorations before the text is drawn.

Instance Methods

backgroundColor

Returns the background color of the text block.

```
- (NSColor *)backgroundColor
```

Return Value
The background color of the text block.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setBackgroundColor:` (page 2769)

Declared In
`NSTextTable.h`

borderColorForEdge:

Returns the border color of the specified text block edge.

```
- (NSColor *)borderColorForEdge:(NSRectEdge)edge
```

Parameters
edge
  The edge of the text block in question.

Return Value
The border color of the text block edge *edge*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setBorderColor:forEdge:` (page 2769)

Declared In
`NSTextTable.h`

boundsRectForContentRect:inRect:textContainer:characterRange:

Returns the rectangle the text in the block actually occupies, including padding, borders, and margins.

```
- (NSRect)boundsRectForContentRect:(NSRect)contentRect inRect:(NSRect)rect
    textContainer:(NSTextContainer *)textContainer characterRange:(NSRange)charRange
```

Parameters

contentRect
The actual rectangle in which the text was laid out, as determined by `rectForLayoutAtPoint:inRect:textContainer:characterRange:` (page 2768).

rect
The initial rectangle in *textContainer* proposed by the typesetter.

textContainer
The text container being used for the layout.

charRange
The range of the characters in the `NSTextStorage` object whose glyphs are to be drawn.

Return Value
The rectangle the text in the block actually occupies, including padding, borders, and margins.

Discussion
This methods is called by the typesetter after the text block is laid out to return the rectangle the text in the block actually occupies, including padding, borders, and margins.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `rectForLayoutAtPoint:inRect:textContainer:characterRange:` (page 2768)

Declared In
`NSTextTable.h`

contentWidth

Returns the width of the text block.

```
- (CGFloat)contentWidth
```

Return Value
The width of the text block. This value must be interpreted according to the value type returned by `contentWidthValueType` (page 2767).

Discussion
This is a convenience method that invokes `valueForDimension:NSTextBlockWidth`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setContentWidth:type:` (page 2770)
- `contentWidthValueType` (page 2767)

Declared In
`NSTextTable.h`

contentWidthValueType

Returns the type of value stored for the text block width.

```
- (NSTextBlockValueType)contentWidthValueType
```

Return Value
The value type for the text block width. This determines how the width value should be interpreted.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setContentWidth:type:` (page 2770)
- `contentWidth` (page 2766)

Declared In
`NSTextTable.h`

drawBackgroundWithFrame:inView:characterRange:layoutManager:

Called by the layout manager to draw any colors and other decorations before the text is drawn.

```
- (void)drawBackgroundWithFrame:(NSRect)frameRect inView:(NSView *)controlView
    characterRange:(NSRange)charRange layoutManager:(NSLayoutManager *)layoutManager
```

Parameters

frameRect
The bounds rectangle in view coordinates.

controlView
The view in which drawing occurs.

charRange
The range of the characters in the `NSTextStorage` object whose glyphs are to be drawn.

layoutManager
The layout manager controlling the typesetting.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTextTable.h`

init

Initializes and returns an empty text block object.

```
- (id)init
```

Return Value
An initialized text block object.

Discussion
This is the designated initializer for `NSTextBlock`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTextTable.h`

rectForLayoutAtPoint:inRect:textContainer:characterRange:

Returns the rectangle within which glyphs should be laid out for the specified arguments.

```
- (NSRect)rectForLayoutAtPoint:(NSPoint)startingPoint inRect:(NSRect)rect
    textContainer:(NSTextContainer *)textContainer characterRange:(NSRange)charRange
```

Parameters

startingPoint
The location, in container coordinates, where layout begins.

rect
The rectangle in which the block is constrained to lie. For top-level blocks, this is the container rectangle of *textContainer*; for nested blocks, this is the layout rectangle of the enclosing block.

textContainer
The text container being used for the layout.

charRange
The range of the characters in the `NSTextStorage` object whose glyphs are to be drawn.

Return Value
The rectangle within which glyphs should be laid out.

Discussion
This method is called by the typesetter before the text block is laid out to return the rectangle within which glyphs should be laid out.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- boundsRectForContentRect:inRect:textContainer:characterRange:` (page 2766)

Declared In
`NSTextTable.h`

setBackgroundColor:

Sets the background color of the text block.

```
- (void)setBackgroundColor:(NSColor *)color
```

Parameters

color

The new background color.

Availability
Available in Mac OS X v10.4 and later.

See Also
- backgroundColor (page 2765)

Declared In
NSTextTable.h

setBorderColor:

Sets the color of all borders of the text block.

```
- (void)setBorderColor:(NSColor *)color
```

Parameters

color

The new color.

Discussion
This setting has no visible effect unless the border width is larger than the default, which is 0.

Availability
Available in Mac OS X v10.4 and later.

See Also
- borderColorForEdge: (page 2765)
- setBorderColor:forEdge: (page 2769)
- setWidth:type:forLayer: (page 2771)

Related Sample Code
iSpend

Declared In
NSTextTable.h

setBorderColor:forEdge:

Sets the border color of the specified edge of the text block.

```
- (void)setBorderColor:(NSColor *)color forEdge:(NSRectEdge)edge
```

Parameters

color

The new color.

edge

The edge whose color is to be set.

Discussion

This setting has no visible effect unless the border width is larger than the default, which is 0.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `borderColorForEdge:` (page 2765)
- `setWidth:type:forLayer:` (page 2771)

Declared In

`NSTextTable.h`

setContentWidth:type:

Sets the width of the text block.

```
- (void)setContentWidth:(CGFloat)val type:(NSTextBlockValueType)type
```

Parameters

val

The new value for the width.

type

The type of value being provided. This controls how *val* is interpreted.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `contentWidth` (page 2766)
- `contentWidthValueType` (page 2767)

Declared In

`NSTextTable.h`

setValue:type:forDimension:

Sets a dimension of the text block.

```
- (void)setValue:(CGFloat)val type:(NSTextBlockValueType)type
    forDimension:(NSTextBlockDimension)dimension
```

Parameters

val

The new value for the dimension.

type
The type of value being provided. This controls how *val* is interpreted.

dimension
The dimension to set.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `valueForDimension:` (page 2772)
- `valueTypeForDimension:` (page 2773)

Declared In
`NSTextTable.h`

setVerticalAlignment:

Sets the vertical alignment of the text block.

```
- (void)setVerticalAlignment:(NSTextBlockVerticalAlignment)alignment
```

Parameters
alignment
The new vertical alignment for the text block.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `verticalAlignment` (page 2773)

Related Sample Code
iSpend

Declared In
`NSTextTable.h`

setWidth:type:forLayer:

Sets the width of all edges of a specified layer of the text block.

```
- (void)setWidth:(CGFloat)val type:(NSTextBlockValueType)type
    forLayer:(NSTextBlockLayer)layer
```

Parameters
val
The new value for the specified edge width.

type
The type of value being provided. This controls how *val* is interpreted.

layer
The layer of the text block to modify.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `widthForLayer:edge:` (page 2774)
- `setWidth:type:forLayer:edge:` (page 2772)

Related Sample Code
iSpend

Declared In
`NSTextTable.h`

setWidth:type:forLayer:edge:

Sets the width of a specified edge of a specified layer of the text block.

```
- (void)setWidth:(CGFloat)val type:(NSTextBlockValueType)type
    forLayer:(NSTextBlockLayer)layer edge:(NSRectEdge)edge
```

Parameters

val
The new value for the specified edge width.

type
The type of value being provided. This controls how *val* is interpreted.

layer
The layer of the text block to modify.

edge
The edge of the layer to modify.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setWidth:type:forLayer:` (page 2771)
- `widthForLayer:edge:` (page 2774)

Related Sample Code
iSpend

Declared In
`NSTextTable.h`

valueForDimension:

Returns the value of the specified text block dimension.

```
- (CGFloat)valueForDimension:(NSTextBlockDimension)dimension
```

Return Value
The value for the specified dimension. This value should be interpreted according to the value type returned by `valueTypeForDimension:` (page 2773).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setValue:type:forDimension:` (page 2770)
`valueTypeForDimension:` (page 2773)

Declared In
`NSTextTable.h`

valueTypeForDimension:

Returns the value type of the specified text block dimension.

```
- (NSTextBlockValueType)valueTypeForDimension:(NSTextBlockDimension)dimension
```

Return Value
The value type for the specified text block dimension. This result determines how the value for the dimension should be interpreted.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setValue:type:forDimension:` (page 2770)
`valueForDimension:` (page 2772)

Declared In
`NSTextTable.h`

verticalAlignment

Returns the vertical alignment of the text block.

```
- (NSTextBlockVerticalAlignment)verticalAlignment
```

Return Value
The vertical alignment of the text block.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setVerticalAlignment:` (page 2771)

Declared In
`NSTextTable.h`

widthForLayer:edge:

Returns the width of an edge of a specified layer of the text block.

```
- (CGFloat)widthForLayer:(NSTextBlockLayer)layer edge:(NSRectEdge)edge
```

Parameters

layer

The layer to examine.

edge

The edge of the layer to examine.

Return Value

The width of the *edge* of *layer*. This value must be interpreted according to the value type returned by `widthValueTypeForLayer:edge:` (page 2774).

Availability

Available in Mac OS X v10.4 and later.

See Also

`widthValueTypeForLayer:edge:` (page 2774)

`- setWidth:type:forLayer:edge:` (page 2772)

`- setWidth:type:forLayer:` (page 2771)

Declared In

`NSTextTable.h`

widthValueTypeForLayer:edge:

Returns the value type of an edge of a specified layer of the text block.

```
- (NSTextBlockValueType)widthValueTypeForLayer:(NSTextBlockLayer)layer
    edge:(NSRectEdge)edge
```

Parameters

layer

The layer to examine.

edge

The edge of the layer to examine.

Return Value

The value type of the *edge* of *layer*. This determines how the value for this *edge* of *layer* should be interpreted.

Availability

Available in Mac OS X v10.4 and later.

See Also

`widthForLayer:edge:` (page 2774)

`- setWidth:type:forLayer:edge:` (page 2772)

Declared In

`NSTextTable.h`

Constants

Text Block Value Type Constants

The following constants specify values used by the methods `setValue:type:forDimension:` (page 2770) and `valueTypeForDimension:` (page 2773)

```
enum {
    NSTextBlockAbsoluteValueType    = 0,
    NSTextBlockPercentageValueType  = 1
};
typedef NSUInteger NSTextBlockValueType;
```

Constants

`NSTextBlockAbsoluteValueType`

Absolute value in points.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockPercentageValueType`

Percentage value (out of 100).

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockValueType`

A type defined for the text block value type constants.

Text Block Dimension Constants

The following constants specify values used by the methods `setValue:type:forDimension:` (page 2770), `valueForDimension:` (page 2772), and `valueTypeForDimension:` (page 2773).

```
enum {
    NSTextBlockWidth            = 0,
    NSTextBlockMinimumWidth     = 1,
    NSTextBlockMaximumWidth     = 2,
    NSTextBlockHeight           = 4,
    NSTextBlockMinimumHeight    = 5,
    NSTextBlockMaximumHeight    = 6
};
typedef NSUInteger NSTextBlockDimension;
```

Constants

`NSTextBlockWidth`

Width of the text block.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockMinimumWidth`
Minimum width of the text block.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockMaximumWidth`
Maximum width of the text block.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockHeight`
Height of the text block.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockMinimumHeight`
Minimum height of the text block.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockMaximumHeight`
Maximum height of the text block.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockDimension`
A type defined for the text block dimension constants.

Text Block Layer Constants

The following constants specify values used by the methods `setContentWidth:type:` (page 2770), `contentWidthValueType` (page 2767), `setWidth:type:forLayer:edge:` (page 2772), `setWidth:type:forLayer:` (page 2771), `widthForLayer:edge:` (page 2774), and `widthValueTypeForLayer:edge:` (page 2774).

```
enum {
    NSTextBlockPadding  = -1,
    NSTextBlockBorder   =  0,
    NSTextBlockMargin   =  1
};
typedef NSInteger NSTextBlockLayer;
```

Constants

`NSTextBlockPadding`
Padding of the text block: space surrounding the content area extending to the border.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockBorder`
Border of the text block: space between padding and margin, typically colored to present a visible boundary.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockMargin`
Margin of the text block: space surrounding the border.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockLayer`
A type defined for the text block layer constants.

Text Block Vertical Alignment Constants

The following constants specify values used by the methods `setVerticalAlignment:` (page 2771) and `verticalAlignment` (page 2773).

```
enum {
    NSTextBlockTopAlignment         = 0,
    NSTextBlockMiddleAlignment      = 1,
    NSTextBlockBottomAlignment      = 2,
    NSTextBlockBaselineAlignment    = 3
};
typedef NSUInteger NSTextBlockVerticalAlignment;
```

Constants

`NSTextBlockTopAlignment`
Aligns adjacent blocks at their top.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockMiddleAlignment`
Aligns adjacent blocks at their middle.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockBottomAlignment`
Aligns adjacent blocks at their bottom.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockBaselineAlignment`
Aligns adjacent blocks at the baseline of the first line of text in the block.

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextBlockVerticalAlignment`
A type defined for the text block vertical alignment constants.

NSTextContainer Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTextContainer.h
Companion guide	Text System Overview
Related sample code	Quartz Composer WWDC 2005 TextEdit QuickLookSketch Sketch+Accessibility Sketch-112 TextSizingExample

Overview

An `NSTextContainer` object defines a region where text is laid out. An NSLayoutManager uses NSTextContainers to determine where to break lines, lay out portions of text, and so on. NSTextContainer defines rectangular regions, but you can create subclasses that define regions of other shapes, such as circular regions, regions with holes in them, or regions that flow alongside graphics.

Adopted Protocols

NSCoding

- `- encodeWithCoder:`
- `- initWithCoder:`

Tasks

Creating an Instance

- `initWithContainerSize:` (page 2782)
  Initializes a text container with a specified bounding rectangle.

Managing Text Components

- `setLayoutManager:` (page 2787)
  Sets the receiver's layout manager.
- `layoutManager` (page 2784)
  Returns the receiver's layout manager.
- `replaceLayoutManager:` (page 2785)
  Replaces the layout manager for the group of text system objects containing the receiver.
- `setTextView:` (page 2788)
  Sets the receiver's text view.
- `textView` (page 2789)
  Returns the receiver's text view.

Controlling Size

- `setContainerSize:` (page 2786)
  Sets the size of the receiver's bounding rectangle.
- `containerSize` (page 2781)
  Returns the size of the receiver's bounding rectangle, regardless of the size of its region.
- `setWidthTracksTextView:` (page 2789)
  Controls whether the receiver adjusts the width of its bounding rectangle when its text view is resized.
- `widthTracksTextView` (page 2790)
  Returns whether the receiver adjusts the width of its bounding rectangle when its text view is resized.
- `setHeightTracksTextView:` (page 2786)
  Controls whether the receiver adjusts the height of its bounding rectangle when its text view is resized.
- `heightTracksTextView` (page 2782)
  Returns whether the receiver adjusts the height of its bounding rectangle when its text view is resized.

Setting Line Fragment Padding

- `setLineFragmentPadding:` (page 2788)
  Sets the amount by which text is inset within line fragment rectangles.
- `lineFragmentPadding` (page 2784)
  Returns the amount by which text is inset within line fragment rectangles.

Calculating Text Layout

`- lineFragmentRectForProposedRect:sweepDirection:movementDirection:remainingRect:` (page 2785)

Overridden by subclasses to calculate and return the longest rectangle available in the proposed rectangle for displaying text, or `NSZeroRect` if there is none according to the receiver's region definition.

`- isSimpleRectangularTextContainer` (page 2783)

Overridden by subclasses to return whether the receiver's region is a rectangle with no holes or gaps and whose edges are parallel to the text view's coordinate system axes.

Mouse Hit Testing

`- containsPoint:` (page 2781)

Overridden by subclasses to return whether a point lies within the receiver's region or on the region's edge—not simply within its bounding rectangle.

Instance Methods

containerSize

Returns the size of the receiver's bounding rectangle, regardless of the size of its region.

```
- (NSSize)containerSize
```

Return Value
The size of the receiver's bounding rectangle, regardless of the size of its region.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- textContainerInset` (page 2956) (NSTextView)
`- setContainerSize:` (page 2786)

Related Sample Code
TextLayoutDemo

Declared In
`NSTextContainer.h`

containsPoint:

Overridden by subclasses to return whether a point lies within the receiver's region or on the region's edge—not simply within its bounding rectangle.

```
- (BOOL)containsPoint:(NSPoint)aPoint
```

Parameters

aPoint

The point in question.

Return Value

YES if *aPoint* lies within the receiver's region or on the region's edge—not simply within its bounding rectangle—NO otherwise.

Discussion

For example, if the receiver defines a donut shape and *aPoint* lies in the hole, this method returns NO. This method can be used for hit testing of mouse events.

NSTextContainer's implementation merely checks that *aPoint* lies within its bounding rectangle.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSTextContainer.h

heightTracksTextView

Returns whether the receiver adjusts the height of its bounding rectangle when its text view is resized.

```
- (BOOL)heightTracksTextView
```

Return Value

YES if the receiver adjusts the height of its bounding rectangle when its text view is resized, NO otherwise.

Discussion

If the receiver does track the text view height, its height is adjusted to the height of the text view minus twice the inset height (as given by NSTextView's `textContainerInset` (page 2956) method).

See *Text System Storage Layer Overview* for more information on size tracking.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `widthTracksTextView` (page 2790)
- `setHeightTracksTextView:` (page 2786)

Declared In

NSTextContainer.h

initWithContainerSize:

Initializes a text container with a specified bounding rectangle.

```
- (id)initWithContainerSize:(NSSize)aSize
```

Parameters

aSize

The size of the text container's bounding rectangle.

Return Value

The newly initialized text container.

Discussion

The new text container must be added to an `NSLayoutManager` object before it can be used. The text container must also have an `NSTextView` object set for text to be displayed. This method is the designated initializer for the `NSTextContainer` class.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addTextContainer:` (page 1449) (NSLayoutManager)
- `setTextView:` (page 2788)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

QuickLookSketch

Sketch+Accessibility

Sketch-112

TextViewConfig

Declared In

`NSTextContainer.h`

isSimpleRectangularTextContainer

Overridden by subclasses to return whether the receiver's region is a rectangle with no holes or gaps and whose edges are parallel to the text view's coordinate system axes.

```
- (BOOL)isSimpleRectangularTextContainer
```

Return Value

`YES` if the receiver's region is a rectangle with no holes or gaps and whose edges are parallel to the text view's coordinate system axes, `NO` otherwise.

Discussion

A text container whose shape changes can return `YES` if its region is currently a simple rectangle, but when its shape does change it must send `textContainerChangedGeometry:` (page 1518) to its layout manager so the layout can be recalculated.

`NSTextContainer`'s implementation of this method returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

TextLayoutDemo

Declared In
NSTextContainer.h

layoutManager

Returns the receiver's layout manager.

```
- (NSLayoutManager *)layoutManager
```

Return Value
The text container's layout manager.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setLayoutManager: (page 2787)
- replaceLayoutManager: (page 2785)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSTextContainer.h

lineFragmentPadding

Returns the amount by which text is inset within line fragment rectangles.

```
- (CGFloat)lineFragmentPadding
```

Return Value
The amount by which text is inset within line fragment rectangles, in points.

Availability
Available in Mac OS X v10.0 and later.

See Also
- lineFragmentRectForProposedRect:sweepDirection:movementDirection:remainingRect: (page 2785)
- setLineFragmentPadding: (page 2788)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextSizingExample

Declared In
NSTextContainer.h

lineFragmentRectForProposedRect:sweepDirection:movementDirection:remainingRect:

Overridden by subclasses to calculate and return the longest rectangle available in the proposed rectangle for displaying text, or `NSZeroRect` if there is none according to the receiver's region definition.

```
- (NSRect)lineFragmentRectForProposedRect:(NSRect)proposedRect
    sweepDirection:(NSLineSweepDirection)sweepDirection
    movementDirection:(NSLineMovementDirection)movementDirection
    remainingRect:(NSRectPointer)remainingRect
```

Parameters

proposedRect

The proposed rectangle in which to layout text.

sweepDirection

The line sweep direction.

movementDirection

The line movement direction.

remainingRect

Upon return, the unused, possibly shifted, portion of *proposedRect* that's available for further text, or `NSZeroRect` if there is no remainder.

Return Value

The longest rectangle available in the proposed rectangle for displaying text, or `NSZeroRect` if there is none according to the receiver's region definition.

Discussion

There is no guarantee as to the width of the proposed rectangle or to its location. For example, the proposed rectangle is likely to be much wider than the width of the receiver. The receiver should examine *proposedRect* to see that it intersects its bounding rectangle and should return a modified rectangle based on *sweepDirection* and *movementDirection*, whose possible values are listed in the class description. If *sweepDirection* is `NSLineSweepRight`, for example, the receiver uses this information to trim the right end of *proposedRect* as needed rather than the left end.

If *proposedRect* doesn't completely overlap the region along the axis of *movementDirection* and *movementDirection* isn't `NSLineDoesntMove`, this method can either shift the rectangle in that direction as much as needed so that it does completely overlap, or return `NSZeroRect` to indicate that the proposed rectangle simply doesn't fit.

See the class description for more information on overriding this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextContainer.h`

replaceLayoutManager:

Replaces the layout manager for the group of text system objects containing the receiver.

```
- (void)replaceLayoutManager:(NSLayoutManager *)aLayoutManager
```

Parameters

aLayoutManager

The new layout manager.

Discussion

All text containers and text views sharing the original layout manager share the new layout manager. This method makes all the adjustments necessary to keep these relationships intact, unlike `setLayoutManager:` (page 2787).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `layoutManager` (page 2784)

Declared In

`NSTextContainer.h`

setContainerSize:

Sets the size of the receiver's bounding rectangle.

```
- (void)setContainerSize:(NSSize)aSize
```

Parameters

aSize

The new size of the text container's bounding rectangle.

Discussion

This method also sends `textContainerChangedGeometry:` (page 1518) to the text container's layout manager.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setTextContainerInset:` (page 2946) (NSTextView)
- `containerSize` (page 2781)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit
QuickLookSketch
Sketch+Accessibility
Sketch-112
TextSizingExample

Declared In

`NSTextContainer.h`

setHeightTracksTextView:

Controls whether the receiver adjusts the height of its bounding rectangle when its text view is resized.

```
- (void)setHeightTracksTextView:(BOOL)flag
```

Parameters

flag

YES if the receiver should follow changes to the height of its text view, NO otherwise.

Discussion

See *Text System Storage Layer Overview* for more information on size tracking.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setContainerSize: (page 2786)
- setWidthTracksTextView: (page 2789)
- heightTracksTextView (page 2782)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility
Sketch-112
TextLayoutDemo
TextSizingExample

Declared In

NSTextContainer.h

setLayoutManager:

Sets the receiver's layout manager.

```
- (void)setLayoutManager:(NSLayoutManager *)aLayoutManager
```

Parameters

aLayoutManager

The new layout manager.

Discussion

This method is invoked automatically when you add a text container to a layout manager; you should never need to invoke it directly, but might want to override it. If you want to replace the layout manager for an established group of text system objects, use replaceLayoutManager: (page 2785).

Availability

Available in Mac OS X v10.0 and later.

See Also

- addTextContainer: (page 1449) (NSLayoutManager)
- layoutManager (page 2784)

Declared In

NSTextContainer.h

setLineFragmentPadding:

Sets the amount by which text is inset within line fragment rectangles.

```
- (void)setLineFragmentPadding:(CGFloat)aFloat
```

Parameters

aFloat

The amount by which text is inset within line fragment rectangles, in points.

Discussion

This method also sends `textContainerChangedGeometry:` (page 1518) to the text container's layout manager.

Line fragment padding is not designed to express text margins. Instead, use the `NSTextView` method `setTextContainerInset:` (page 2946), paragraph margin attributes, or the position of the text view within a superview.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- lineFragmentRectForProposedRect:sweepDirection:movementDirection:remainingRect:` (page 2785)

`- lineFragmentPadding` (page 2784)

Related Sample Code

TipWrapper

Declared In

`NSTextContainer.h`

setTextView:

Sets the receiver's text view.

```
- (void)setTextView:(NSTextView *)aTextView
```

Parameters

aTextView

The new text view.

Discussion

This method sends `setTextContainer:` (page 2945) to *aTextView* to complete the association of the text container and text view.

Because you usually specify a text container when you create a text view, you should rarely need to invoke this method. A text container doesn't need a text view to calculate line fragment rectangles, but must have one to display text.

You can use this method to disconnect a text view from a group of text system objects by sending this message to its text container and passing `nil` as *aTextView*.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `initWithFrame:textContainer:` (page 2896) (NSTextView)
- `replaceTextContainer:` (page 2916) (NSTextView)

Related Sample Code
Sketch-112

Declared In
`NSTextContainer.h`

setWidthTracksTextView:

Controls whether the receiver adjusts the width of its bounding rectangle when its text view is resized.

```
- (void)setWidthTracksTextView:(BOOL)flag
```

Parameters

flag

`YES` if the receiver should follow changes to the width of its text view, `NO` otherwise.

Discussion
See *Text System Storage Layer Overview* for more information on size tracking.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setContainerSize:` (page 2786)
- `setHeightTracksTextView:` (page 2786)
- `widthTracksTextView` (page 2790)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility
Sketch-112
TextLayoutDemo
TextSizingExample

Declared In
`NSTextContainer.h`

textView

Returns the receiver's text view.

```
- (NSTextView *)textView
```

Return Value
The receiver's text view, or `nil` if it has none.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTextView:` (page 2788)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSTextContainer.h`

widthTracksTextView

Returns whether the receiver adjusts the width of its bounding rectangle when its text view is resized.

```
- (BOOL)widthTracksTextView
```

Return Value
`YES` if the receiver adjusts the width of its bounding rectangle when its text view is resized, `NO` otherwise.

Discussion
If the receiver does track the text view width, its width is adjusted to the width of the text view minus twice the inset width (as given by `NSTextView`'s `textContainerInset` (page 2956) method).

See *Text System Storage Layer Overview* for more information on size tracking.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `heightTracksTextView` (page 2782)
- `setWidthTracksTextView:` (page 2789)

Declared In
`NSTextContainer.h`

Constants

NSLineSweepDirection

These constants describe the progression of text on a page. The typesetter decides which way text is supposed to flow and passes these values as arguments to the text container, which uses them to calculate the next line rectangle.

```
typedef enum {
    NSLineSweepLeft  = 0,
    NSLineSweepRight = 1,
    NSLineSweepDown  = 2,
    NSLineSweepUp    = 3
} NSLineSweepDirection;
```

Constants

`NSLineSweepLeft`

Characters move from right to left.

Available in Mac OS X v10.0 and later.

Declared in `NSTextContainer.h`.

`NSLineSweepRight`

Characters move from left to right.

Available in Mac OS X v10.0 and later.

Declared in `NSTextContainer.h`.

`NSLineSweepDown`

Characters move from top to bottom.

Available in Mac OS X v10.0 and later.

Declared in `NSTextContainer.h`.

`NSLineSweepUp`

Characters move from bottom to top.

Available in Mac OS X v10.0 and later.

Declared in `NSTextContainer.h`.

Discussion

Line sweep is the direction text progresses within a line. See *Text System Storage Layer Overview*.

The only values currently used by the supplied typesetters are `NSLineSweepRight` and `NSLineMovesDown`. An NSTextContainer subclass should be prepared to deal with any value, and an NSTypesetter subclass should be able to use any of them.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextContainer.h`

NSLineMovementDirection

Line movement is the direction in which lines move. See *Text System Storage Layer Overview*.

```
typedef enum {
    NSLineDoesntMove = 0,
    NSLineMovesLeft  = 1,
    NSLineMovesRight = 2,
    NSLineMovesDown  = 3,
    NSLineMovesUp    = 4
} NSLineMovementDirection;
```

Constants

`NSLineMovesLeft`

Lines move from right to left.

Available in Mac OS X v10.0 and later.

Declared in `NSTextContainer.h`.

`NSLineMovesRight`

Lines move from left to right.

Available in Mac OS X v10.0 and later.

Declared in `NSTextContainer.h`.

`NSLineMovesDown`

Lines move from top to bottom.

Available in Mac OS X v10.0 and later.

Declared in `NSTextContainer.h`.

`NSLineMovesUp`

Lines move from bottom to top.

Available in Mac OS X v10.0 and later.

Declared in `NSTextContainer.h`.

`NSLineDoesntMove`

Line has no movement.

Available in Mac OS X v10.0 and later.

Declared in `NSTextContainer.h`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextContainer.h`

NSTextField Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTextField.h
Companion guide	Text Fields
Related sample code	EnhancedAudioBurn EnhancedDataBurn FunHouse ImageClient OpenALExample

Overview

An `NSTextField` object is a kind of `NSControl` that displays text that the user can select or edit and that sends its action message to its target when the user presses the Return key while editing.

The `NSTextField` class uses the `NSTextFieldCell` class to implement its user interface.

Tasks

Controlling Editability and Selectability

`- setEditable:` (page 2804)
Controls whether the user can edit the receiver's text.

`- isEditable` (page 2799)
Returns a Boolean value indicating whether the user is allowed to select and edit the receiver's text.

`- setSelectable:` (page 2806)
Sets whether the receiver is selectable (but not editable).

- `isSelectable` (page 2800)
  Returns a Boolean value indicating whether the user is allowed to select the receiver's text.

Controlling Rich Text Behavior

- `setAllowsEditingTextAttributes:` (page 2801)
  Controls whether the receiver allows the user to change font attributes of the receiver's text.
- `allowsEditingTextAttributes` (page 2796)
  Returns a Boolean value indicating whether the user is allowed to change font attributes of the receiver's text.
- `setImportsGraphics:` (page 2805)
  Controls whether the receiver allows the user to drag image files into it.
- `importsGraphics` (page 2798)
  Returns a Boolean value indicating whether the receiver allows the user to drag image files into it.

Setting the Text Color

- `setTextColor:` (page 2806)
  Sets the color used to draw the receiver's text.
- `textColor` (page 2807)
  Returns the color used to draw the receiver's text.

Controlling the Background

- `setBackgroundColor:` (page 2801)
  Sets the color of the background that the receiver's cell draws behind the text.
- `backgroundColor` (page 2797)
  Returns the color of the background that the receiver's cell draws behind the text.
- `setDrawsBackground:` (page 2803)
  Controls whether the receiver's cell draws its background color behind its text.
- `drawsBackground` (page 2798)
  Returns a Boolean value indicating whether the receiver's cell draws its background color.

Setting a Border

- `setBezeled:` (page 2802)
  Controls whether the receiver draws a bezeled border around its contents.
- `isBezeled` (page 2798)
  Returns a Boolean value indicating whether the receiver draws a bezeled frame.
- `setBezelStyle:` (page 2802)
  Sets the receiver's bezel style.
- `bezelStyle` (page 2797)
  Returns the receiver's bezel style.

- `setBordered:` (page 2803)
  Controls whether the receiver draws a solid black border around its contents.
- `isBordered` (page 2799)
  Returns a Boolean value indicating whether the receiver draws a black border around its contents.

Selecting the Text

- `selectText:` (page 2801)
  Ends editing and selects the entire contents of the receiver if it's selectable.

Working with the Responder Chain

- `acceptsFirstResponder` (page 2796)
  Returns a Boolean value indicating whether the receiver is editable.
- `nextText` (page 2800)
  Returns the object selected when the user presses Tab. (Deprecated. Use the `NSView` method `nextKeyView` (page 3192) instead.)
- `previousText` (page 2800)
  Returns the object selected when the user presses Shift-Tab. (Deprecated. Use the `NSView` method `previousKeyView` (page 3197) instead.)
- `setNextText:` (page 2805)
  Sets the object selected when the user presses Tab. (Deprecated. Use the `NSView` method `setNextKeyView:` (page 3226) instead.)
- `setPreviousText:` (page 2805)
  Sets the object selected when the user presses Shift-Tab. (Deprecated. Use the `NSView` method `setNextKeyView:` (page 3226) instead.)

Using Keyboard Interface Control

- `setTitleWithMnemonic:` (page 2807)
  Sets the receiver's string value, using the embedded character as the keyboard mnemonic.

Setting the Delegate

- `setDelegate:` (page 2803)
  Sets the receiver's delegate.
- `delegate` (page 2797)
  Returns the receiver's delegate.

NSText Delegate Method Implementations

- `textShouldBeginEditing:` (page 2809)
  Requests permission to begin editing a text object.

- `textDidBeginEditing:` (page 2807)
  Posts a notification that the text is about to begin editing to the default notification center.
- `textDidChange:` (page 2808)
  Posts a notification that the text has changed and forwards this message to the receiver's cell if it responds.
- `textShouldEndEditing:` (page 2810)
  Performs validation on the receiver's new value.
- `textDidEndEditing:` (page 2808)
  Handles an end of editing.

Instance Methods

acceptsFirstResponder

Returns a Boolean value indicating whether the receiver is editable.

```
- (BOOL)acceptsFirstResponder
```

Return Value
`YES` if the receiver is editable, `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextField.h`

allowsEditingTextAttributes

Returns a Boolean value indicating whether the user is allowed to change font attributes of the receiver's text.

```
- (BOOL)allowsEditingTextAttributes
```

Return Value
`YES` if the receiver allows the user to change font attributes of the receiver's text, otherwise `NO`. You can change text attributes programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `importsGraphics` (page 2798)
- `setAllowsEditingTextAttributes:` (page 2801)

Declared In
`NSTextField.h`

backgroundColor

Returns the color of the background that the receiver's cell draws behind the text.

```
- (NSColor *)backgroundColor
```

Return Value
The color used to draw the background.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawsBackground` (page 2798)
- `setBackgroundColor:` (page 2801)

Declared In
`NSTextField.h`

bezelStyle

Returns the receiver's bezel style.

```
- (NSTextFieldBezelStyle)bezelStyle
```

Return Value
A constant indicating the bezel style. Possible values are described in the `Bezel Styles` (page 522) section of NSTextFieldCell Class Reference (page 2811).

Availability
Available in Mac OS X v10.2 and later.

See Also
- `setBezelStyle:` (page 2802)

Declared In
`NSTextField.h`

delegate

Returns the receiver's delegate.

```
- (id < NSTextFieldDelegate >)delegate
```

Return Value
The object that acts as the receiver's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDelegate:` (page 2803)

Declared In
NSTextField.h

drawsBackground

Returns a Boolean value indicating whether the receiver's cell draws its background color.

```
- (BOOL)drawsBackground
```

Return Value
YES if the receiver's cell draws its background color behind its text, NO if it draws no background.

Discussion
In order to prevent inconsistent rendering, background color rendering is disabled for rounded-bezel text fields.

Availability
Available in Mac OS X v10.0 and later.

See Also
- backgroundColor (page 2797)
- drawsBackground (page 2814) (NSTextFieldCell)
- setDrawsBackground: (page 2803)

Declared In
NSTextField.h

importsGraphics

Returns a Boolean value indicating whether the receiver allows the user to drag image files into it.

```
- (BOOL)importsGraphics
```

Return Value
YES if the receiver allows the user to drag image files into it, otherwise NO. You can add images programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- allowsEditingTextAttributes (page 2796)
- importsGraphics (page 2895) (NSTextView)
- setImportsGraphics: (page 2805)

Declared In
NSTextField.h

isBezeled

Returns a Boolean value indicating whether the receiver draws a bezeled frame.

```
- (BOOL)isBezeled
```

Return Value
YES if the receiver draws a bezeled frame around its contents; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isBordered (page 2799)
- setBezeled: (page 2802)

Declared In
NSTextField.h

isBordered

Returns a Boolean value indicating whether the receiver draws a black border around its contents.

```
- (BOOL)isBordered
```

Return Value
YES if the receiver draws a solid black border around its contents; otherwise NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isBezeled (page 2798)
- setBordered: (page 2803)

Declared In
NSTextField.h

isEditable

Returns a Boolean value indicating whether the user is allowed to select and edit the receiver's text.

```
- (BOOL)isEditable
```

Return Value
YES if the user is allowed to select and edit the receiver's text, NO if the user isn't allowed to edit it (though the user may be able to select it).

Availability
Available in Mac OS X v10.0 and later.

See Also
- isSelectable (page 2800)
- setEditable: (page 2804)

Declared In
NSTextField.h

isSelectable

Returns a Boolean value indicating whether the user is allowed to select the receiver's text.

```
- (BOOL)isSelectable
```

Return Value
`YES` if the user is allowed to select the receiver's text; otherwise `NO`.

Discussion
Selectable text isn't necessarily editable; use `isEditable` (page 2799) to check for editability.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setSelectable:` (page 2806)

Declared In
`NSTextField.h`

nextText

Returns the object selected when the user presses Tab. (Available in Mac OS X v10.0 through Mac OS X v10.1. Use the `NSView` method `nextKeyView` (page 3192) instead.)

```
- (id)nextText
```

Return Value
The object to be selected.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared In
`NSTextField.h`

previousText

Returns the object selected when the user presses Shift-Tab. (Available in Mac OS X v10.0 through Mac OS X v10.1. Use the `NSView` method `previousKeyView` (page 3197) instead.)

```
- (id)previousText
```

Return Value
The object to be selected.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared In
`NSTextField.h`

selectText:

Ends editing and selects the entire contents of the receiver if it's selectable.

```
- (void)selectText:(id)sender
```

Parameters

sender

The sender of the message.

Discussion

If the receiver isn't in some window's view hierarchy, this method has no effect.

Availability

Available in Mac OS X v10.0 and later.

See Also

- isSelectable (page 2800)

Related Sample Code

QTMetadataEditor

TipWrapper

Declared In

NSTextField.h

setAllowsEditingTextAttributes:

Controls whether the receiver allows the user to change font attributes of the receiver's text.

```
- (void)setAllowsEditingTextAttributes:(BOOL)flag
```

Parameters

flag

If YES, the user is permitted to change font attributes of the receiver's text; if *flag* is NO, the user isn't so permitted. You can change text attributes programmatically regardless of this setting.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setImportsGraphics: (page 2805)
- allowsEditingTextAttributes (page 2796)

Declared In

NSTextField.h

setBackgroundColor:

Sets the color of the background that the receiver's cell draws behind the text.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Parameters

aColor

The color used to draw the background.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDrawsBackground:` (page 2803)
- `backgroundColor` (page 2797)

Declared In
`NSTextField.h`

setBezeled:

Controls whether the receiver draws a bezeled border around its contents.

```
- (void)setBezeled:(BOOL)flag
```

Parameters

flag

If `YES`, it draws a bezeled border and invokes `setDrawsBackground:` (page 2803) with an argument of `NO`; if `NO`, the receiver does not draw a border.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isBezeled` (page 2798)
- `setBordered:` (page 2803)

Related Sample Code
FunHouse
Quartz Composer QCTV
Vertex Optimization

Declared In
`NSTextField.h`

setBezelStyle:

Sets the receiver's bezel style.

```
- (void)setBezelStyle:(NSTextFieldBezelStyle)style
```

Parameters

style

A constant indicating the bezel style. Possible values for *style* are described in the `Bezel Styles` (page 522) section of NSTextFieldCell Class Reference (page 2811). You must have already sent the receiver `setBezeled:` (page 2802) with an argument of `YES`

Availability
Available in Mac OS X v10.2 and later.

See Also
- `bezelStyle` (page 2797)

Declared In
`NSTextField.h`

setBordered:

Controls whether the receiver draws a solid black border around its contents.

```
- (void)setBordered:(BOOL)flag
```

Parameters

flag
If `YES`, the receiver draws a border; if `NO`, it draws no border.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isBordered` (page 2799)
- `setBezeled:` (page 2802)

Declared In
`NSTextField.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSTextFieldDelegate >)anObject
```

Parameters

anObject
The object that acts as the receiver's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `delegate` (page 2797)

Declared In
`NSTextField.h`

setDrawsBackground:

Controls whether the receiver's cell draws its background color behind its text.

```
- (void)setDrawsBackground:(BOOL)flag
```

Parameters

flag

If YES, the receiver's cell draws its background; if NO, it draws nothing behind its text.

Discussion

In order to prevent inconsistent rendering, background color rendering is disabled for rounded-bezel text fields.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setBackgroundColor: (page 2801)
- setDrawsBackground: (page 2817) (NSTextFieldCell)
- drawsBackground (page 2798)

Related Sample Code

FunHouse

Quartz Composer QCTV

Declared In

NSTextField.h

setEditable:

Controls whether the user can edit the receiver's text.

```
- (void)setEditable:(BOOL)flag
```

Parameters

flag

If YES, then the user is allowed to both select and edit text. If *flag* is NO, then the user isn't permitted to edit text, and the receiver's selectability is restored to its previous value.

Discussion

For example, if an NSTextField object is selectable but not editable, then made editable for a time, then made not editable, it remains selectable. To guarantee that text is neither editable nor selectable, simply use setSelectable: (page 2806) to turn off selectability.

Availability

Available in Mac OS X v10.0 and later.

See Also

- isEditable (page 2799)

Related Sample Code

FinalCutPro_AppleEvents

FunHouse

SharedMemory

UIElementInspector

With and Without Bindings

Declared In
NSTextField.h

setImportsGraphics:

Controls whether the receiver allows the user to drag image files into it.

```
- (void)setImportsGraphics:(BOOL)flag
```

Parameters

flag

If YES, the receiver accepts dragged images; if NO, it doesn't. You can add images programmatically regardless of this setting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setAllowsEditingTextAttributes: (page 2801)
- setImportsGraphics: (page 2936) (NSTextView)
- importsGraphics (page 2798)

Declared In
NSTextField.h

setNextText:

Sets the object selected when the user presses Tab. (Available in Mac OS X v10.0 through Mac OS X v10.1. Use the NSView method setNextKeyView: (page 3226) instead.)

```
- (void)setNextText:(id)anObject
```

Parameters

anObject

The object to be selected.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared In
NSTextField.h

setPreviousText:

Sets the object selected when the user presses Shift-Tab. (Available in Mac OS X v10.0 through Mac OS X v10.1. Use the NSView method setNextKeyView: (page 3226) instead.)

```
- (void)setPreviousText:(id)anObject
```

Parameters

anObject

The object to be selected.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.1.

Declared In
NSTextField.h

setSelectable:

Sets whether the receiver is selectable (but not editable).

```
- (void)setSelectable:(BOOL)flag
```

Parameters

flag

If YES, the receiver is made selectable but not editable (use setEditable: (page 2804) to make text both selectable and editable). If NO, the text is neither editable nor selectable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setEditable: (page 2804)

Related Sample Code
Quartz Composer QCTV
With and Without Bindings

Declared In
NSTextField.h

setTextColor:

Sets the color used to draw the receiver's text.

```
- (void)setTextColor:(NSColor *)aColor
```

Parameters

aColor

The color used to draw text.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setBackgroundColor: (page 2801)
- setTextColor: (page 2818) (NSTextFieldCell)
- textColor (page 2807)

Related Sample Code
IdentitySample
UIElementInspector
Vertex Optimization

Declared In
NSTextField.h

setTitleWithMnemonic:

Sets the receiver's string value, using the embedded character as the keyboard mnemonic.

```
- (void)setTitleWithMnemonic:(NSString *)aString
```

Parameters

aString

The string to set as the title. The first character preceded by an ampersand ('&') is used as the mnemonic (the first ampersand character is stripped out).

Discussion
Mnemonics are not supported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextField.h

textColor

Returns the color used to draw the receiver's text.

```
- (NSColor *)textColor
```

Return Value
The color used to draw text.

Availability
Available in Mac OS X v10.0 and later.

See Also
- backgroundColor (page 2797)
- textColor (page 2819) (NSTextFieldCell)
- setTextColor: (page 2806)

Declared In
NSTextField.h

textDidBeginEditing:

Posts a notification that the text is about to begin editing to the default notification center.

```
- (void)textDidBeginEditing:(NSNotification *)aNotification
```

Parameters

aNotification

The NSControlTextDidBeginEditingNotification (page 847) notification to post.

Discussion
This action causes the receiver's delegate to receive a `controlTextDidBeginEditing:` (page 846) message. See the `NSControl` class specification for more information on the text delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textDidBeginEditing:` (page 2807)
- `textDidChange:` (page 2808)
- `textShouldEndEditing:` (page 2810)
- `textDidEndEditing:` (page 2808)

Declared In
`NSTextField.h`

textDidChange:

Posts a notification that the text has changed and forwards this message to the receiver's cell if it responds.

```
- (void)textDidChange:(NSNotification *)aNotification
```

Parameters
aNotification
The `NSControlTextDidChangeNotification` (page 848) notification that is posted to the default notification center.

Discussion
This method causes the receiver's delegate to receive a `controlTextDidChange:` (page 846) message. See the `NSControl` class specification for more information on the text delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textShouldBeginEditing:` (page 2809)
- `textDidBeginEditing:` (page 2807)
- `textShouldEndEditing:` (page 2810)
- `textDidEndEditing:` (page 2808)

Declared In
`NSTextField.h`

textDidEndEditing:

Handles an end of editing.

```
- (void)textDidEndEditing:(NSNotification *)aNotification
```

Parameters
aNotification
The notification that editing has ended.

Discussion
After validating the new value, posts an `NSControlTextDidEndEditingNotification` (page 848) to the default notification center. This posting causes the receiver's delegate to receive a `controlTextDidEndEditing:` (page 847) message. After this message, sends `endEditing:` (page 555) to the receiver' cell and handles the key that caused editing to end:

- If the user ended editing by pressing Return, this method tries to send the receiver's action to its target; if unsuccessful, it sends `performKeyEquivalent:` (page 3195) to its `NSView` (for example, to handle the default button on a panel); if that also fails, the receiver simply selects its text.
- If the user ended editing by pressing Tab or Shift-Tab, the receiver tries to have its `NSWindow` object select its next or previous key view, using the `NSWindow` method `selectKeyViewFollowingView:` (page 3364) or `selectKeyViewPrecedingView:` (page 3364). If unsuccessful in doing this, the receiver simply selects its text.

See the `NSControl` class specification for more information on the text delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textShouldBeginEditing:` (page 2809)
- `textDidBeginEditing:` (page 2807)
- `textDidChange:` (page 2808)
- `textShouldEndEditing:` (page 2810)

Declared In
`NSTextField.h`

textShouldBeginEditing:

Requests permission to begin editing a text object.

```
- (BOOL)textShouldBeginEditing:(NSText *)textObject
```

Parameters
textObject
The object to begin editing.

Return Value
`YES` if editing should be allowed to occur, `NO` otherwise.

Discussion
If the receiver isn't editable, returns `NO` immediately. If it is editable and its delegate responds to `control:textShouldBeginEditing:`, this invokes that method and returns the result. Otherwise it simply returns `YES` to allow editing to occur. See the `NSControl` class specification for more information on the text delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textDidBeginEditing:` (page 2807)

- textDidChange: (page 2808)
- textShouldEndEditing: (page 2810)
- textDidEndEditing: (page 2808)

Declared In
NSTextField.h

textShouldEndEditing:

Performs validation on the receiver's new value.

```
- (BOOL)textShouldEndEditing:(NSText *)textObject
```

Parameters

textObject
The text object requesting permission to end editing.

Return Value
YES if the new value is valid; otherwise NO.

Discussion
This method validates the receiver's new value using the NSCell method isEntryAcceptable: (page 568). If the new value is valid and the delegate responds to control:textShouldEndEditing:, invokes that method and returns the result, in addition beeping if the delegate returns NO. See the NSControl class specification for more information on the text delegate method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- textShouldBeginEditing: (page 2809)
- textDidBeginEditing: (page 2807)
- textDidChange: (page 2808)
- textDidEndEditing: (page 2808)

Declared In
NSTextField.h

NSTextFieldCell Class Reference

Inherits from	NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTextFieldCell.h
Companion guide	Text Fields
Related sample code	AnimatedTableView FunHouse PhotoSearch SourceView STUCAuthoringDeviceCocoaSample

Overview

The `NSTextFieldCell` class adds to the text display capabilities of the `NSCell` class by allowing you to set the color of both the text and its background. You can also specify whether the cell draws its background at all.

All of the methods declared by this class are also declared by the `NSTextField` class, which uses `NSTextFieldCell` objects to draw and edit text. These `NSTextField` cover methods call the corresponding `NSTextFieldCell` methods.

Placeholder strings, set using `setPlaceholderString:` (page 2818) or `setPlaceholderAttributedString:` (page 2817), now appear in the text field cell if the actual string is `nil` or `@""`. They are drawn in grey on the cell and are not archived in the “pre-10.2” nib format.

Designated Initializers

When subclassing `NSTextFieldCell` you must implement all of the designated initializers. Those methods are: `initWithCoder:`, `initTextCell:` (page 564), and `initImageCell:` (page 563).

Tasks

Setting the Text Color

- `setTextColor:` (page 2818)
  Sets the color used to draw the receiver's text.
- `textColor` (page 2819)
  Returns the color used to draw the receiver's text.

Setting the Bezel Style

- `setBezelStyle:` (page 2816)
  Sets the receiver's bezel style.
- `bezelStyle` (page 2814)
  Returns the receiver's bezel style.

Controlling the Background

- `setBackgroundColor:` (page 2816)
  Sets the color of the background that the receiver draws behind the text.
- `backgroundColor` (page 2813)
  Returns the color of the background the receiver draws behind the text.
- `setDrawsBackground:` (page 2817)
  Controls whether the receiver draws its background color behind its text.
- `drawsBackground` (page 2814)
  Returns a Boolean value that indicates whether the receiver draws its background color.

Managing the Field Editor

- `setUpFieldEditorAttributes:` (page 2819)
  Sets up the field editor. You never invoke this method directly; by overriding it, however, you can customize the field editor.
- `setWantsNotificationForMarkedText:` (page 2819)
  Directs the cell's associated field editor to post text change notifications.

Managing Placeholder Strings

- `setPlaceholderString:` (page 2818)
  Sets the placeholder of the cell as a plain text string.
- `placeholderString` (page 2815)
  Returns the cell's plain text placeholder string.

- `setPlaceholderAttributedString:` (page 2817)
  Sets the placeholder of the cell as an attributed string.
- `placeholderAttributedString` (page 2815)
  Returns the cell's attributed placeholder string.

Accessing Input Source Locales

- `allowedInputSourceLocales` (page 2813)
  Returns an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.
- `setAllowedInputSourceLocales:` (page 2816)
  Sets an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.

Instance Methods

allowedInputSourceLocales

Returns an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.

```
- (NSArray *)allowedInputSourceLocales
```

Return Value
The locale identifiers of allowed input sources.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setAllowedInputSourceLocales:` (page 2816)

Declared In
`NSTextFieldCell.h`

backgroundColor

Returns the color of the background the receiver draws behind the text.

```
- (NSColor *)backgroundColor
```

Return Value
The background color of the text field cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
- drawsBackground (page 2814)
- backgroundColor (page 2797) (NSTextField)
- setBackgroundColor: (page 2816)

Related Sample Code
DragNDropOutlineView
EnhancedDataBurn
ImageBackground
QTKitMovieShuffler
STUCAuthoringDeviceCocoaSample

Declared In
NSTextFieldCell.h

bezelStyle

Returns the receiver's bezel style.

```
- (NSTextFieldBezelStyle)bezelStyle
```

Return Value
A constant indicating the bezel style of the text field cell. See NSTextFieldBezelStyle (page 2820).

Availability
Available in Mac OS X v10.2 and later.

See Also
- setBezelStyle: (page 2816)

Declared In
NSTextFieldCell.h

drawsBackground

Returns a Boolean value that indicates whether the receiver draws its background color.

```
- (BOOL)drawsBackground
```

Return Value
YES if the receiver draws its background color behind its text, NO if it draws no background.

Discussion
In order to prevent inconsistent rendering, background color rendering is disabled for rounded-bezel text fields.

Availability
Available in Mac OS X v10.0 and later.

See Also
- backgroundColor (page 2813)
- drawsBackground (page 2798) (NSTextField)

- setDrawsBackground: (page 2817)

Related Sample Code
DragNDropOutlineView
EnhancedDataBurn
ImageBackground
QTKitMovieShuffler
STUCAuthoringDeviceCocoaSample

Declared In
NSTextFieldCell.h

placeholderAttributedString

Returns the cell's attributed placeholder string.

```
- (NSAttributedString *)placeholderAttributedString
```

Return Value
The attributed string used as the cell's placeholder.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setPlaceholderAttributedString: (page 2817)
- placeholderString (page 2815)

Declared In
NSTextFieldCell.h

placeholderString

Returns the cell's plain text placeholder string.

```
- (NSString *)placeholderString
```

Return Value
The cell''s placeholder string.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setPlaceholderString: (page 2818)
- placeholderAttributedString (page 2815)

Declared In
NSTextFieldCell.h

setAllowedInputSourceLocales:

Sets an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.

```
- (void)setAllowedInputSourceLocales:(NSArray *)localeIdentifiers
```

Parameters

localeIdentifiers

The new locale identifiers of allowed input sources.

Discussion

You can use the meta-locale identifier, `NSAllRomanInputSourcesLocaleIdentifier` (page 2972), to specify input sources that are limited for Roman script editing.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `allowedInputSourceLocales` (page 2813)

Declared In

`NSTextFieldCell.h`

setBackgroundColor:

Sets the color of the background that the receiver draws behind the text.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Parameters

aColor

The background color of the text field cell.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setDrawsBackground:` (page 2817)
- `setBackgroundColor:` (page 2801) (`NSTextField`)
- `backgroundColor` (page 2813)

Declared In

`NSTextFieldCell.h`

setBezelStyle:

Sets the receiver's bezel style.

```
- (void)setBezelStyle:(NSTextFieldBezelStyle)style
```

Parameters

style

A constant specifying the bezel style of the text field cell. See `NSTextFieldBezelStyle` (page 2820).

Discussion

To set the bezel style, you must have already sent `setBezeled:` (page 2802) with an argument of `YES`.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `bezelStyle` (page 2814)

Declared In

`NSTextFieldCell.h`

setDrawsBackground:

Controls whether the receiver draws its background color behind its text.

```
- (void)setDrawsBackground:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver draws its background; if `NO` it draws nothing behind its text.

Discussion

In order to prevent inconsistent rendering, background color rendering is disabled for rounded-bezel text fields.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setBackgroundColor:` (page 2816)
- `setDrawsBackground:` (page 2803) (`NSTextField`)
- `drawsBackground` (page 2814)

Declared In

`NSTextFieldCell.h`

setPlaceholderAttributedString:

Sets the placeholder of the cell as an attributed string.

```
- (void)setPlaceholderAttributedString:(NSAttributedString *)string
```

Return Value

The attributed string to use as a placeholder.

Discussion

Note that invoking this successfully will clear out any plain text string set by `setPlaceholderString:` (page 2818).

Availability
Available in Mac OS X v10.3 and later.

See Also
- `placeholderAttributedString` (page 2815)
- `setPlaceholderString:` (page 2818)

Declared In
`NSTextFieldCell.h`

setPlaceholderString:

Sets the placeholder of the cell as a plain text string.

```
- (void)setPlaceholderString:(NSString *)string
```

Parameters

string
The placeholder string.

Discussion
Note that invoking this successfully will clear out any attributed string set by `setPlaceholderAttributedString:` (page 2817).

Availability
Available in Mac OS X v10.3 and later.

See Also
- `placeholderString` (page 2815)
- `setPlaceholderAttributedString:` (page 2817)

Declared In
`NSTextFieldCell.h`

setTextColor:

Sets the color used to draw the receiver's text.

```
- (void)setTextColor:(NSColor *)aColor
```

Parameters

aColor
The color used to draw the text.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBackgroundColor:` (page 2816)
- `setTextColor:` (page 2806) (`NSTextField`)
- `textColor` (page 2819)

Declared In
NSTextFieldCell.h

setUpFieldEditorAttributes:

Sets up the field editor. You never invoke this method directly; by overriding it, however, you can customize the field editor.

```
- (NSText *)setUpFieldEditorAttributes:(NSText *)textObj
```

Discussion
When you override this method, you should generally invoke the implementation of `super` and return the *`textObj`* argument. For information on field editors, see "Using the Window's Field Editor".

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextFieldCell.h

setWantsNotificationForMarkedText:

Directs the cell's associated field editor to post text change notifications.

```
- (void)setWantsNotificationForMarkedText:(BOOL)flag
```

Parameters
`flag`
If `YES`, the field editor posts text change notifications (`NSTextDidChangeNotification`) while editing marked text; if `NO`, notifications are delayed until the marked text confirmation.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTextFieldCell.h

textColor

Returns the color used to draw the receiver's text.

```
- (NSColor *)textColor
```

Return Value
The color used to draw the text.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `backgroundColor` (page 2813)
- `textColor` (page 2807) (`NSTextField`)

- setTextColor: (page 2818)

Related Sample Code
PhotoSearch

Declared In
NSTextFieldCell.h

Constants

NSTextFieldBezelStyle

Specify the bezel style of the text field cell.

```
enum {
    NSTextFieldSquareBezel  = 0,
    NSTextFieldRoundedBezel = 1
};
typedef NSUInteger NSTextFieldBezelStyle;
```

Constants

NSTextFieldSquareBezel
Corners are square.
Available in Mac OS X v10.2 and later.
Declared in NSTextFieldCell.h.

NSTextFieldRoundedBezel
Corners are rounded.
Available in Mac OS X v10.2 and later.
Declared in NSTextFieldCell.h.

Discussion
The bezel style of a text field is set using the bezelStyle (page 2814) and setBezelStyle: (page 2816) methods.

Availability
Available in Mac OS X v10.2 and later.

Declared In
NSTextFieldCell.h

NSTextInputContext Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	NSTextInputContext.h

Overview

An `NSTextInputContext` object represents the Cocoa text input system. The text input system communicates primarily with the client of the activated input context via the `NSTextInputClient` protocol.

Tasks

Creating an Input Context

`- initWithClient:` (page 2826)
The designated initializer

Getting the Input Context and Client

`+ currentInputContext` (page 2824)
Returns the current, activated, text input context object.

`client` (page 2823) *property*
The owner of this input context. (read-only)

Configuring the Input Context

`acceptsGlyphInfo` (page 2822) *property*
A Boolean value that indicates whether the client handles `NSGlyphInfoAttributeName` or not.

`allowedInputSourceLocales` (page 2823) *property*
The set of keyboard input source locales allowed when this input context is active.

Activating the Input Context

- `activate` (page 2824)
  Activates the receiver.
- `deactivate` (page 2825)
  Deactivates the receiver.

Handling Input Sources

- `handleEvent:` (page 2825)
  Tells the Cocoa text input system to handle mouse or key events.
- `discardMarkedText` (page 2825)
  Tells the Cocoa test input system to discard the current conversion session.
- `invalidateCharacterCoordinates` (page 2826)
  Notifies the Cocoa text input system that the position information previously queried via methods like `firstRectForCharacterRange:actualRange:` needs to be updated.

`keyboardInputSources` (page 2823) *property*
The array of keyboard text input source identifier strings available to the receiver. (read-only)

`selectedKeyboardInputSource` (page 2823) *property*
The identifier string for the selected keyboard text input source.

+ `localizedNameForInputSource:` (page 2824)
  Returns the display name for the given text input source identifier.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

acceptsGlyphInfo

A Boolean value that indicates whether the client handles `NSGlyphInfoAttributeName` or not.

```
@property BOOL acceptsGlyphInfo;
```

Discussion
The default value is determined by examining the return value from sending a `validAttributesForMarkedText` message to the client at initialization.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTextInputContext.h`

allowedInputSourceLocales

The set of keyboard input source locales allowed when this input context is active.

```
@property(copy) NSArray *allowedInputSourceLocales;
```

Discussion
`NSAllRomanInputSourcesLocaleIdentifier` can be specified as a valid locale.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTextInputContext.h`

client

The owner of this input context. (read-only)

```
@property(readonly) id <NSTextInputClient> client;
```

Discussion
The client (owner) of the input context, typically an `NSView` instance, retains its `NSTextInputContext` instance. The `NSTextInputContext` instance doesn't retain its client.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTextInputContext.h`

keyboardInputSources

The array of keyboard text input source identifier strings available to the receiver. (read-only)

```
@property(readonly) NSArray *keyboardInputSources;
```

Discussion
The Text Input Source Services API identifies text input sources with text input source identifier strings (for example, `com.apple.inputmethod.Kotoeri.Japanese`) supplied by the underlying text input sources framework. The ID corresponds to the `kTISPropertyInputSourceID` attribute.

For more information on the Text Input Source Services API, see *Text Input Source Services Reference*.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTextInputContext.h`

selectedKeyboardInputSource

The identifier string for the selected keyboard text input source.

```
@property(copy) NSString *selectedKeyboardInputSource;
```

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSTextInputContext.h

Class Methods

currentInputContext

Returns the current, activated, text input context object.

```
+ (NSTextInputContext *)currentInputContext;
```

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSTextInputContext.h

localizedNameForInputSource:

Returns the display name for the given text input source identifier.

```
+ (NSString *)localizedNameForInputSource:(NSString *)inputSourceIdentifier;
```

Parameters

inputSourceIdentifier
The text input source identifier.

Return Value
The localized display name for *inputSourceIdentifier*.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSTextInputContext.h

Instance Methods

activate

Activates the receiver.

```
- (void)activate;
```

Discussion
You should not call this method directly; it is invoked by the system. It is provided as an override point for subclasses.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `deactivate` (page 2825)

Declared In
`NSTextInputContext.h`

deactivate

Deactivates the receiver.

```
- (void)deactivate;
```

Discussion
You should not call this method directly; it is invoked by the system. It is provided as an override point for subclasses.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `activate` (page 2824)

Declared In
`NSTextInputContext.h`

discardMarkedText

Tells the Cocoa test input system to discard the current conversion session.

```
- (void)discardMarkedText;
```

Discussion
The client should clear its marked range when sending this message.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTextInputContext.h`

handleEvent:

Tells the Cocoa text input system to handle mouse or key events.

```
- (BOOL)handleEvent:(NSEvent *)theEvent;
```

Parameters

theEvent

The event to handle.

Return Value

`YES` if the system consumed the event; otherwise `NO`.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSTextInputContext.h`

initWithClient:

The designated initializer

```
- (id)initWithClient:(id <NSTextInputClient>)theClient;
```

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSTextInputContext.h`

invalidateCharacterCoordinates

Notifies the Cocoa text input system that the position information previously queried via methods like `firstRectForCharacterRange:actualRange:` needs to be updated.

```
- (void)invalidateCharacterCoordinates;
```

Availability

Available in Mac OS X v10.6 and later.

Related Sample Code

TextInputView

Declared In

`NSTextInputContext.h`

Notifications

NSTextInputContextKeyboardSelectionDidChangeNotification

Posted after the selected text input source changes.

Availability

Available in Mac OS X v10.6 and later.

Declared In
`NSTextInputContext.h`

NSTextList Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSTextList.h
Availability	Available in Mac OS X v10.4 and later.

Overview

An `NSTextList` object represents a section of text that forms a single list. The visible elements of the list, including list markers, appear in the text as they do for lists created by hand. The list object, however, allows the list to be recognized as such by the text system. This enables automatic creation of markers and spacing. Text lists are used in HTML import and export.

Text lists appear as attributes on paragraphs, as part of the paragraph style. An NSParagraphStyle may have an array of text lists, representing the nested lists containing the paragraph, in order from outermost to innermost. For example, if list1 contains four paragraphs, the middle two of which are also in the inner list2, then the text lists array for the first and fourth paragraphs is (list1), while the text lists array for the second and third paragraphs is (list1, list2).

The methods implementing this are `textLists` (page 1877) on `NSParagraphStyle`, and `setTextLists:` (page 1726) on `NSMutableParagraphStyle`.

In addition, `NSAttributedString` has convenience methods for lists:`rangeOfTextList:atIndex:` (page 267), which determines the range covered by a list, and `itemNumberInTextList:atIndex:` (page 265), which determines the ordinal position within a list of a particular item.

Tasks

Creating a Text List

`- initWithMarkerFormat:options:` (page 2830)
Returns an initialized text list.

Working with Markers

- `markerFormat` (page 2832)
  Returns the marker format string used by the receiver.
- `markerForItemNumber:` (page 2831)
  Returns the computed value for a specific ordinal position in the list.

Getting List Options

- `listOptions` (page 2831)
  Returns the list options mask value of the receiver.

Managing Item Numbering

- `startingItemNumber` (page 2832)
  Returns the starting item number for the text list.
- `setStartingItemNumber:` (page 2832)
  Sets the starting item number for the text list.

Instance Methods

initWithMarkerFormat:options:

Returns an initialized text list.

```
- (id)initWithMarkerFormat:(NSString *)format options:(NSUInteger)mask
```

Parameters

format
The marker format for the text list.

mask
The marker options for the text list. Values for *mask* are listed in "Constants" (page 2833).

Return Value
An initialized text list.

Discussion
The marker format is specified as a constant string, except for a numbering specifier, which takes the form {*keyword*}. The currently supported values for *keyword* include:

```
box
check
circle
diamond
disc
hyphen
```

```
square
lower-hexadecimal
upper-hexadecimal
octal
lower-alpha or lower-latin
upper-alpha or upper-latin
lower-roman
upper-roman
decimal
```

Thus, for example, `@"({decimal})"` would specify the format for a list numbered (1), (2), (3), and so on, and `@"{upper-roman}"` would specify the format for a list numbered I, II, III, IV, and so on. (All of these keywords are included in the Cascading Style Sheets level 3 specification.)

Availability
Available in Mac OS X v10.4 and later.

See Also
- `markerFormat` (page 2832)
- `listOptions` (page 2831)

Declared In
`NSTextList.h`

listOptions

Returns the list options mask value of the receiver.

```
- (NSUInteger)listOptions
```

Return Value
The list options mask value of the receiver.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTextList.h`

markerForItemNumber:

Returns the computed value for a specific ordinal position in the list.

```
- (NSString *)markerForItemNumber:(NSInteger)itemNum
```

Parameters
itemNum
The ordinal position in the list whose computed marker value is desired.

Return Value
The computed maker value for *itemNum*.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTextList.h

markerFormat

Returns the marker format string used by the receiver.

```
- (NSString *)markerFormat
```

Return Value
The marker format string used by the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- initWithMarkerFormat:options: (page 2830)

Declared In
NSTextList.h

setStartingItemNumber:

Sets the starting item number for the text list.

```
- (void)setStartingItemNumber:(NSInteger)itemNum
```

Parameters
itemNum
The item number.

Discussion
The default value is 1. This value will be used only for ordered lists, and ignored in other cases.

Availability
Available in Mac OS X v10.6 and later.

See Also
- startingItemNumber (page 2832)

Declared In
NSTextList.h

startingItemNumber

Returns the starting item number for the text list.

```
- (NSInteger)startingItemNumber
```

Return Value
The item number.

Discussion
The default value is `1`. This value will be used only for ordered lists, and ignored in other cases.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setStartingItemNumber:` (page 2832)

Declared In
`NSTextList.h`

Constants

The following constant specifies an option mask used with `initWithMarkerFormat:options:` (page 2830).

Constant	Description
`NSTextListPrepend-EnclosingMarker`	Specifies that a nested list should include the marker for its enclosing superlist before its own marker. Available in Mac OS X v10.4 and later. Declared in `NSTextList.h`.

NSTextStorage Class Reference

Inherits from	NSMutableAttributedString : NSAttributedString : NSObject
Conforms to	NSCoding (NSAttributedString) NSCopying (NSAttributedString) NSMutableCopying (NSAttributedString) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTextStorage.h AppKit/NSTextStorageScripting.h
Companion guides	Text System Overview Text System Storage Layer Overview Cocoa Scripting Guide
Related sample code	Quartz Composer WWDC 2005 TextEdit QuickLookSketch Sketch+Accessibility Sketch-112 TextSizingExample

Overview

`NSTextStorage` is a semiconcrete subclass of `NSMutableAttributedString` that manages a set of client `NSLayoutManager` objects, notifying them of any changes to its characters or attributes so that they can relay and redisplay the text as needed. `NSTextStorage` defines the fundamental storage mechanism of the Application Kit's extended text-handling system.

`NSTextStorage` also defines a set of methods, listed under "Getting and setting scriptable properties" in the Method Types section, useful for getting and setting scriptable properties of `NSTextStorage` objects. Unless you are dealing with scriptability, you do not normally need to invoke these methods directly. In particular, using the `characters`, `words` or `paragraphs` methods or their corresponding setter methods is an inefficient way to manipulate the text storage, since these methods create and return many objects. Instead, use the text access methods defined by `NSMutableAttributedString`, `NSAttributedString`, `NSMutableString`, and `NSString` to perform character-level manipulation.

Tasks

Managing NSLayoutManager Objects

- `addLayoutManager:` (page 2837)
  Adds a layout manager to the receiver's set of layout managers.
- `removeLayoutManager:` (page 2844)
  Removes a layout manager from the receiver's set of layout managers.
- `layoutManagers` (page 2843)
  Returns the receiver's layout managers.

Handling Text Edited Messages

- `edited:range:changeInLength:` (page 2839)
  Tracks changes made to the receiver, allowing the text storage to record the full extent of changes made.
- `ensureAttributesAreFixedInRange:` (page 2841)
  Ensures that attributes are fixed in the given range.
- `fixesAttributesLazily` (page 2842)
  Returns whether the receiver fixes attributes lazily.
- `invalidateAttributesInRange:` (page 2843)
  Invalidates attributes in the specified range.
- `processEditing` (page 2844)
  Cleans up changes made to the receiver and notifies its delegate and layout managers of changes.

Determining the Nature of Changes

- `editedMask` (page 2840)
  Returns the kinds of edits pending for the receiver

Determining the Extent of Changes

- `editedRange` (page 2841)
  Returns the range of the receiver to which pending changes have been made, whether of characters or of attributes.
- `changeInLength` (page 2838)
  Returns the difference between the current length of the edited range and its length before editing began.

Setting the Delegate

- `setDelegate:` (page 2845)
  Sets the receiver's delegate.
- `delegate` (page 2839)
  Returns the receiver's delegate.

Getting and Setting Scriptable Properties

- `attributeRuns` (page 2838)
  Returns an array of the receiver's attribute runs.
- `setAttributeRuns:` (page 2845)
  Sets the receiver's attribute runs.
- `characters` (page 2839)
  Returns the receiver's text as an array of characters.
- `setCharacters:` (page 2845)
  Sets the text storage's text.
- `font` (page 2842)
  Returns the receiver's font.
- `setFont:` (page 2846)
  Sets the text storage's font.
- `foregroundColor` (page 2842)
  Returns the text storage's foreground color.
- `setForegroundColor:` (page 2846)
  Sets the text storage's foreground color.
- `paragraphs` (page 2843)
  Returns an array of the text storage's paragraphs.
- `setParagraphs:` (page 2846)
  Sets the text storage's paragraphs.
- `words` (page 2847)
  Returns an array of the text storage's words.
- `setWords:` (page 2847)
  Sets the text storage's words.

Instance Methods

addLayoutManager:

Adds a layout manager to the receiver's set of layout managers.

```
- (void)addLayoutManager:(NSLayoutManager *)aLayoutManager
```

Parameters

aLayoutManager

The layout manager to add.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `removeLayoutManager:` (page 2844)
- `layoutManagers` (page 2843)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
QuickLookSketch
Sketch+Accessibility
Sketch-112
TextViewConfig

Declared In
`NSTextStorage.h`

attributeRuns

Returns an array of the receiver's attribute runs.

```
- (NSArray *)attributeRuns
```

Return Value
An array of the receiver's attribute runs.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextStorageScripting.h`

changeInLength

Returns the difference between the current length of the edited range and its length before editing began.

```
- (NSInteger)changeInLength
```

Return Value
The difference between the current length of the edited range and its length before editing began. That is, before the receiver was sent the first `beginEditing` message or a single `edited:range:changeInLength:` (page 2839) message.

Discussion
This difference is accumulated with each invocation of `edited:range:changeInLength:` (page 2839), until a final message processes the changes.

The receiver's delegate and layout managers can use this information to determine the nature of edits in their respective notification methods.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `editedRange` (page 2841)
- `editedMask` (page 2840)

Declared In
`NSTextStorage.h`

characters

Returns the receiver's text as an array of characters.

```
- (NSArray *)characters
```

Special Considerations

Do not use this method unless you are dealing directly with scriptability. For indexed access to characters, use `NSAttributedString`'s `length` method to access the string, and `NSString`'s `characterAtIndex:` method to access the individual characters.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextStorageScripting.h`

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDelegate:` (page 2845)

Declared In
`NSTextStorage.h`

edited:range:changeInLength:

Tracks changes made to the receiver, allowing the text storage to record the full extent of changes made.

```
- (void)edited:(NSUInteger)mask range:(NSRange)oldRange
    changeInLength:(NSInteger)lengthChange
```

Parameters

mask

A mask specifying the nature of the changes. The value is made by combining with the C bitwise OR operator the options described in "Change notifications" (page 2847).

oldRange

The extent of characters affected before the change took place.

lengthChange

The number of characters added to or removed from *oldRange*. If no characters where edited as noted by *mask*, its value is irrelevant and undefined. For example, when replacing "The" with "Several" in the string "The files couldn't be saved", *oldRange* is {0, 3} and *lengthChange* is 4.

Discussion

This method invokes `processEditing` (page 2844). `NSTextStorage` invokes this method automatically each time it makes a change to its attributed string. Subclasses that override or add methods that alter their attributed strings directly should invoke this method after making those changes; otherwise you should not invoke this method. The information accumulated with this method is then used in an invocation of `processEditing` (page 2844) to report the affected portion of the receiver.

The methods for querying changes, `editedRange` (page 2841) and `changeInLength` (page 2838), indicate the extent of characters affected after the change. This method expects the characters before the change because that information is readily available as the argument to whatever method performs the change (such as `replaceCharactersInRange:withString:`).

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ClipboardViewer

Declared In

`NSTextStorage.h`

editedMask

Returns the kinds of edits pending for the receiver

```
- (NSUInteger)editedMask
```

Return Value

A mask describing the kinds of edits pending for the receiver.

Discussion

Use the C bitwise AND operator to test the mask; testing for equality will fail if additional mask flags are added later. The receiver's delegate and layout managers can use this information to determine the nature of edits in their respective notification methods.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `editedRange` (page 2841)
- `changeInLength` (page 2838)

Declared In
NSTextStorage.h

editedRange

Returns the range of the receiver to which pending changes have been made, whether of characters or of attributes.

```
- (NSRange)editedRange
```

Return Value
The range of the receiver to which pending changes have been made, whether of characters or of attributes.

Discussion
The receiver's delegate and layout managers can use this information to determine the nature of edits in their respective notification methods.

Availability
Available in Mac OS X v10.0 and later.

See Also
- changeInLength (page 2838)
- editedMask (page 2840)

Declared In
NSTextStorage.h

ensureAttributesAreFixedInRange:

Ensures that attributes are fixed in the given range.

```
- (void)ensureAttributesAreFixedInRange:(NSRange)range
```

Parameters
range
The range of characters whose attributes might be examined.

Discussion
An NSTextStorage object using lazy attribute fixing is required to call this method before accessing any attributes within *range*. This method gives attribute fixing a chance to occur if necessary. NSTextStorage subclasses wishing to support laziness must call this method from all attribute accessors they implement.

Availability
Available in Mac OS X v10.0 and later.

See Also
- fixesAttributesLazily (page 2842)
- invalidateAttributesInRange: (page 2843)

Declared In
NSTextStorage.h

fixesAttributesLazily

Returns whether the receiver fixes attributes lazily.

```
- (BOOL)fixesAttributesLazily
```

Return Value
`YES` if the text storage fixes attributes lazily, `NO` otherwise.

Discussion
By default, custom `NSTextStorage` subclasses are not lazy, but the provided concrete subclass is lazy by default.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextStorage.h`

font

Returns the receiver's font.

```
- (NSFont *)font
```

Return Value
The receiver's font.

Discussion
In Mac OS X v10.5 and later, if the font has not been set, this method and font-related scripting commands assume Helvetica 12.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextStorageScripting.h`

foregroundColor

Returns the text storage's foreground color.

```
- (NSColor *)foregroundColor
```

Return Value
The text storage's foreground color.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextStorageScripting.h`

invalidateAttributesInRange:

Invalidates attributes in the specified range.

```
- (void)invalidateAttributesInRange:(NSRange)range
```

Parameters

range

The range of characters whose attributes should be invalidated.

Discussion

Called from `processEditing` (page 2844) to invalidate attributes when the text storage changes. If the receiver is not lazy, this method simply calls `fixAttributesInRange:` (page 1709). If lazy attribute fixing is in effect, this method instead records the range needing fixing.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `ensureAttributesAreFixedInRange:` (page 2841)
- `fixesAttributesLazily` (page 2842)

Declared In

`NSTextStorage.h`

layoutManagers

Returns the receiver's layout managers.

```
- (NSArray *)layoutManagers
```

Return Value

The receiver's layout managers.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addLayoutManager:` (page 2837)
- `removeLayoutManager:` (page 2844)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

`NSTextStorage.h`

paragraphs

Returns an array of the text storage's paragraphs.

```
- (NSArray *)paragraphs
```

Return Value
An array of the text storage's paragraphs.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextStorageScripting.h

processEditing

Cleans up changes made to the receiver and notifies its delegate and layout managers of changes.

```
- (void)processEditing
```

Discussion
This method is automatically invoked in response to an `edited:range:changeInLength:` (page 2839) message. You should never need to invoke it directly.

This method begins by posting an `NSTextStorageWillProcessEditingNotification` (page 2848) to the default notification center (which results in the delegate receiving a `textStorageWillProcessEditing:` message). After this, it posts an `NSTextStorageDidProcessEditingNotification` (page 2848) to the default notification center (which results in the delegate receiving a `textStorageDidProcessEditing:` message). Finally, it sends a `textStorage:edited:range:changeInLength:invalidatedRange:` (page 1521) message to each of the receiver's layout managers using the argument values provided.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextStorage.h

removeLayoutManager:

Removes a layout manager from the receiver's set of layout managers.

```
- (void)removeLayoutManager:(NSLayoutManager *)aLayoutManager
```

Parameters
aLayoutManager
The layout manager to remove.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addLayoutManager:` (page 2837)
- `layoutManagers` (page 2843)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
QuickLookSketch
Sketch+Accessibility

Sketch-112

Declared In
NSTextStorage.h

setAttributeRuns:

Sets the receiver's attribute runs.

```
- (void)setAttributeRuns:(NSArray *)attributeRuns
```

Parameters

attributeRuns
The array of attribute runs to set.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextStorageScripting.h

setCharacters:

Sets the text storage's text.

```
- (void)setCharacters:(NSArray *)characters
```

Parameters

characters
The characters to set as the text of the text storage.

Special Considerations

Do not use this method if you are not dealing directly with scriptability. Use `NSMutableAttributedString`'s `mutableString` method to return a string object that will be tracked by the corresponding attributed string for modifications.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextStorageScripting.h

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)anObject
```

Parameters

anObject
The new delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `delegate` (page 2839)

Declared In
`NSTextStorage.h`

setFont:

Sets the text storage's font.

```
- (void)setFont:(NSFont *)font
```

Parameters

font
The new font.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DockTile
SpeedometerView
WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
`NSTextStorageScripting.h`

setForegroundColor:

Sets the text storage's foreground color.

```
- (void)setForegroundColor:(NSColor *)color
```

Parameters

color
The new foreground color.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextStorageScripting.h`

setParagraphs:

Sets the text storage's paragraphs.

```
- (void)setParagraphs:(NSArray *)paragraphs
```

Parameters

paragraphs

An array of strings to set as the text storage's paragraphs.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextStorageScripting.h

setWords:

Sets the text storage's words.

```
- (void)setWords:(NSArray *)words
```

Parameters

words

An array of strings to set as the text storage's words.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextStorageScripting.h

words

Returns an array of the text storage's words.

```
- (NSArray *)words
```

Return Value
An array of the text storage's words.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextStorageScripting.h

Constants

Change notifications

These constants are used in `edited:range:changeInLength:` (page 2839).

```
enum {
   NSTextStorageEditedAttributes = 1,
   NSTextStorageEditedCharacters = 2
};
```

Constants

`NSTextStorageEditedAttributes`

Attributes were added, removed, or changed.

Available in Mac OS X v10.0 and later.

Declared in `NSTextStorage.h`.

`NSTextStorageEditedCharacters`

Characters were added, removed, or replaced.

Available in Mac OS X v10.0 and later.

Declared in `NSTextStorage.h`.

Discussion

These values are also OR'ed together in notifications to inform instances of `NSLayoutManager` was changed—see `textStorage:edited:range:changeInLength:invalidatedRange:` (page 1521).

Declared In

`NSTextStorage.h`

Notifications

NSTextStorageDidProcessEditingNotification

Posted after a text storage finishes processing edits in `processEditing` (page 2844).

Observers other than the delegate shouldn't make further changes to the text storage. The notification object is the text storage object that processed the edits. This notification does not contain a *`userInfo`* dictionary.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextStorage.h`

NSTextStorageWillProcessEditingNotification

Posted before a text storage finishes processing edits in `processEditing` (page 2844).

Observers other than the delegate shouldn't make further changes to the text storage. The notification object is the text storage object that is about to process the edits. This notification does not contain a *`userInfo`* dictionary.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextStorage.h`

NSTextTab Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSParagraphStyle.h
Companion guide	Text System Overview
Related sample code	Quartz Composer WWDC 2005 TextEdit

Overview

An `NSTextTab` object represents a tab in an `NSParagraphStyle` object, storing an alignment type and location. `NSTextTab` objects are most frequently used with the Application Kit's text system and with NSRulerView and NSRulerMarker objects. See the appropriate class specifications for more information on these uses.

The text system supports four alignment types: left, center, right, and decimal (based on the decimal separator character of the locale in effect). These alignment types are absolute, not based on the line sweep direction of text. For example, tabbed text is always positioned to the left of a right-aligned tab, whether the line sweep direction is left to right or right to left. A tab's location, on the other hand, is relative to the back margin. A tab set at 1.5", for example, is at 1.5" from the right in right to left text.

Adopted Protocols

NSCoding

- `- encodeWithCoder:`
- `- initWithCoder:`

NSCopying

- `- copyWithZone:`

Tasks

Creating an NSTextTab

- `initWithType:location:` (page 2851)
  Initializes a newly allocated NSTextTab with an alignment of *type* at *location* on the paragraph.
- `initWithTextAlignment:location:options:` (page 2850)
  Initializes a text tab with the text alignment, location, and options.

Getting Tab Stop Information

- `location` (page 2851)
  Returns the receiver's ruler location relative to the back margin.
- `tabStopType` (page 2852)
  Returns the receiver's tab stop type.

Getting Text Tab Information

- `alignment` (page 2850)
  Returns the text alignment of the receiver.
- `options` (page 2851)
  Returns the dictionary of attributes associated with the receiver.

Instance Methods

alignment

Returns the text alignment of the receiver.

```
- (NSTextAlignment)alignment
```

Return Value
The text alignment of the receiver as an `NSTextAlignment` (page 2747) constant

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSParagraphStyle.h`

initWithTextAlignment:location:options:

Initializes a text tab with the text alignment, location, and options.

```
- (id)initWithTextAlignment:(NSTextAlignment)alignment location:(CGFloat)loc
    options:(NSDictionary *)options
```

Discussion
The text alignment is used to determine the position of text inside the tab column. See `NSTextTabType` (page 2852) for a mapping between alignments and tab stop types

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSParagraphStyle.h`

initWithType:location:

Initializes a newly allocated NSTextTab with an alignment of *type* at *location* on the paragraph.

```
- (id)initWithType:(NSTextTabType)type location:(CGFloat)location
```

Discussion
The location is relative to the back margin, based on the line sweep direction of the paragraph. *type* can be any of the values described in `NSTextTabType` (page 2852).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSParagraphStyle.h`

location

Returns the receiver's ruler location relative to the back margin.

```
- (CGFloat)location
```

Return Value
The receiver's ruler location relative to the back margin.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSParagraphStyle.h`

options

Returns the dictionary of attributes associated with the receiver.

```
- (NSDictionary *)options
```

Return Value
The dictionary of attributes associated with the receiver.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSParagraphStyle.h

tabStopType

Returns the receiver's tab stop type.

```
- (NSTextTabType)tabStopType
```

Return Value
The receiver's tab stop type. Possible values are listed in `NSTextTabType` (page 2852).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSParagraphStyle.h

Constants

NSTextTabType

These constants describe the various type of tab stop.

```
typedef enum _NSTextTabType {
    NSLeftTabStopType = 0,
    NSRightTabStopType,
    NSCenterTabStopType,
    NSDecimalTabStopType
} NSTextTabType;
```

Constants

`NSLeftTabStopType`
A left-aligned tab stop.
Available in Mac OS X v10.0 and later.
Declared in `NSParagraphStyle.h`.

`NSRightTabStopType`
A right-aligned tab stop.
Available in Mac OS X v10.0 and later.
Declared in `NSParagraphStyle.h`.

`NSCenterTabStopType`
A center-aligned tab stop.

Available in Mac OS X v10.0 and later.

Declared in `NSParagraphStyle.h`.

`NSDecimalTabStopType`
Aligns columns of numbers by the decimal point.

Available in Mac OS X v10.0 and later.

Declared in `NSParagraphStyle.h`.

Discussion
The following mappings define the conversions between text alignment in NSTextTab and tab stop types defined by NSTextTab:

Alignment	Tab Stop Type
`NSLeftTextAlignment`	`NSLeftTabStopType`
`NSRightTextAlignment`	`NSRightTabStopType`
`NSCenterTextAlignment`	`NSCenterTabStopType`
`NSJustifiedTextAlignment`	`NSLeftTabStopType`
`NSNaturalTextAlignment`	`NSLeftTabStopType` or `NSRightTabStopType`, depending on the user setting
`NSRightTextAlignment` with terminator	`NSDecimalTabStopType`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSParagraphStyle.h`

Terminating character

This constant specifies the terminating character for a tab column.

```
NSString *NSTabColumnTerminatorsAttributeName;
```

Constants
`NSTabColumnTerminatorsAttributeName`
The value is an `NSCharacterSet` object.

The character set is used to determine the terminating character for a tab column. The tab and newline characters are implied even if they don't exist in the character set. This attribute is optional.

Available in Mac OS X v10.3 and later.

Declared in `NSParagraphStyle.h`.

Declared In
`NSParagraphStyle.h`

NSTextTable Class Reference

Inherits from	NSTextBlock : NSObject
Conforms to	NSCoding (NSTextBlock) NSCopying (NSTextBlock) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSTextTable.h
Availability	Available in Mac OS X v10.4 and later.
Companion guides	Text System Overview Text Layout Programming Guide
Related sample code	iSpend

Overview

An `NSTextTable` object represents a text table as a whole. It is responsible for laying out and drawing the text table blocks it contains, and it maintains the basic parameters of the table.

Tasks

Getting and Setting Number of Columns

- `numberOfColumns` (page 2859)
  Returns the number of columns in the text table.
- `setNumberOfColumns:` (page 2861)
  Sets the number of columns in the text table.

Getting and Setting Layout Algorithm

- `layoutAlgorithm` (page 2858)
  Returns the text table layout algorithm.

- `setLayoutAlgorithm:` (page 2861)
  Sets the text table layout algorithm.

Collapsing Borders

- `collapsesBorders` (page 2857)
  Returns whether the text table borders are collapsible.
- `setCollapsesBorders:` (page 2860)
  Sets whether the text table borders are collapsible.

Hiding Empty Cells

- `hidesEmptyCells` (page 2858)
  Returns whether the text table hides empty cells.
- `setHidesEmptyCells:` (page 2860)
  Sets whether the text table hides empty cells.

Determining Layout Rectangles

- `rectForBlock:layoutAtPoint:inRect:textContainer:characterRange:` (page 2859)
  Returns the rectangle within which glyphs should be laid out for a text table block.
- `boundsRectForBlock:contentRect:inRect:textContainer:characterRange:` (page 2856)
  Returns the rectangle the text table block actually occupies, including padding, borders, and margins.

Drawing the Table

- `drawBackgroundForBlock:withFrame:inView:characterRange:layoutManager:` (page 2857)
  Draws any colors and other decorations for a text table block.

Instance Methods

boundsRectForBlock:contentRect:inRect:textContainer:characterRange:

Returns the rectangle the text table block actually occupies, including padding, borders, and margins.

```
- (NSRect)boundsRectForBlock:(NSTextTableBlock *)block
    contentRect:(NSRect)contentRect inRect:(NSRect)rect
    textContainer:(NSTextContainer *)textContainer characterRange:(NSRange)charRange
```

Parameters

block
  The text table block that wants to determine where to layout its glyphs.

contentRect
The actual rectangle in which the text was laid out, as determined by `rectForLayoutAtPoint:inRect:textContainer:characterRange:` (page 2768).

rect
The initial rectangle in *textContainer* proposed by the typesetter.

textContainer
The text container being used for the layout.

charRange
The range of the characters whose glyphs are to be drawn.

Return Value
The rectangle the text table block actually occupies, including padding, borders, and margins.

Discussion
This method is called by the text table block *block* after it is laid out to determine the rectangle the text table block actually occupies, including padding, borders, and margins.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- rectForBlock:layoutAtPoint:inRect:textContainer:characterRange:` (page 2859)

Declared In
`NSTextTable.h`

collapsesBorders

Returns whether the text table borders are collapsible.

```
- (BOOL)collapsesBorders
```

Return Value
`YES` if the text table borders are collapsible, `NO` otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setCollapsesBorders:` (page 2860)

Declared In
`NSTextTable.h`

drawBackgroundForBlock:withFrame:inView:characterRange:layoutManager:

Draws any colors and other decorations for a text table block.

```
- (void)drawBackgroundForBlock:(NSTextTableBlock *)block withFrame:(NSRect)frameRect
    inView:(NSView *)controlView characterRange:(NSRange)charRange
   layoutManager:(NSLayoutManager *)layoutManager
```

Parameters

block
The text table block that wants to draw its background.

frameRect
The area in which drawing occurs.

controlView
The view controlling the drawing.

charRange
The range of the characters whose glyphs are to be drawn.

layoutManager
The layout manager controlling the typesetting.

Discussion
This methods is called by the text table block *block* to draw any colors and other decorations before the text is drawn.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTextTable.h

hidesEmptyCells

Returns whether the text table hides empty cells.

```
- (BOOL)hidesEmptyCells
```

Return Value
YES if the text table hides empty cells, NO otherwise.

Discussion
If empty cells are hidden, locations with empty cells allow the background of the enclosing block or text container to show through.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setHidesEmptyCells: (page 2860)

Declared In
NSTextTable.h

layoutAlgorithm

Returns the text table layout algorithm.

```
- (NSTextTableLayoutAlgorithm)layoutAlgorithm
```

Return Value
The text table layout algorithm.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setLayoutAlgorithm:` (page 2861)

Declared In
`NSTextTable.h`

numberOfColumns

Returns the number of columns in the text table.

```
- (NSUInteger)numberOfColumns
```

Return Value
The number of columns in the text table.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setNumberOfColumns:` (page 2861)

Declared In
`NSTextTable.h`

rectForBlock:layoutAtPoint:inRect:textContainer:characterRange:

Returns the rectangle within which glyphs should be laid out for a text table block.

```
- (NSRect)rectForBlock:(NSTextTableBlock *)block layoutAtPoint:(NSPoint)startingPoint
    inRect:(NSRect)rect textContainer:(NSTextContainer *)textContainer
   characterRange:(NSRange)charRange
```

Parameters

block
The text table block that wants to determine where to layout its glyphs.

startingPoint
The location, in container coordinates, where layout begins.

rect
The rectangle in which the block is constrained to lie. For top-level blocks, this is the container rectangle of *textContainer*; for nested blocks, this is the layout rectangle of the enclosing block.

textContainer
The text container being used for the layout.

charRange
The range of the characters whose glyphs are to be drawn.

Return Value
The rectangle within which glyphs should be laid out.

Discussion
This method is called by the text table block *block* to determine the rectangle within which glyphs should be laid out for the text table block.

Availability
Available in Mac OS X v10.4 and later.

See Also
- boundsRectForBlock:contentRect:inRect:textContainer:characterRange: (page 2856)

Declared In
NSTextTable.h

setCollapsesBorders:

Sets whether the text table borders are collapsible.

```
- (void)setCollapsesBorders:(BOOL)flag
```

Parameters
flag
YES if the text table borders should be collapsible, NO otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- collapsesBorders (page 2857)

Related Sample Code
iSpend

Declared In
NSTextTable.h

setHidesEmptyCells:

Sets whether the text table hides empty cells.

```
- (void)setHidesEmptyCells:(BOOL)flag
```

Parameters
flag
YES if the text table should hide empty cells, NO otherwise.

Discussion
If empty cells are hidden, locations with empty cells allow the background of the enclosing block or text container to show through.

Availability
Available in Mac OS X v10.4 and later.

See Also
- hidesEmptyCells (page 2858)

Related Sample Code
iSpend

Declared In
NSTextTable.h

setLayoutAlgorithm:

Sets the text table layout algorithm.

```
- (void)setLayoutAlgorithm:(NSTextTableLayoutAlgorithm)algorithm
```

Parameters

algorithm
The new layout algorithm.

Availability
Available in Mac OS X v10.4 and later.

See Also
- layoutAlgorithm (page 2858)

Related Sample Code
iSpend

Declared In
NSTextTable.h

setNumberOfColumns:

Sets the number of columns in the text table.

```
- (void)setNumberOfColumns:(NSUInteger)numCols
```

Parameters

numCols
The new number of columns.

Availability
Available in Mac OS X v10.4 and later.

See Also
- numberOfColumns (page 2859)

Related Sample Code
iSpend

Declared In
NSTextTable.h

Constants

NSTextTableLayoutAlgorithm

These constants, specifying the type of text table layout algorithm, are used with `setLayoutAlgorithm:` (page 2861).

```
enum {
    NSTextTableAutomaticLayoutAlgorithm = 0,
    NSTextTableFixedLayoutAlgorithm     = 1
};
typedef NSUInteger NSTextTableLayoutAlgorithm;
```

Constants

`NSTextTableAutomaticLayoutAlgorithm`

Specifies automatic layout algorithm

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

`NSTextTableFixedLayoutAlgorithm`

Specifies fixed layout algorithm

Available in Mac OS X v10.4 and later.

Declared in `NSTextTable.h`.

Declared In

`NSTextTable.h`

NSTextTableBlock Class Reference

Inherits from	NSTextBlock : NSObject
Conforms to	NSCoding (NSTextBlock) NSCopying (NSTextBlock) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSTextTable.h
Availability	Available in Mac OS X v10.4 and later.
Companion guides	Text System Overview Text Layout Programming Guide
Related sample code	iSpend

Overview

An `NSTextTableBlock` object represents a text block that appears as a cell in a text table.

Tasks

Creation

`- initWithTable:startingRow:rowSpan:startingColumn:columnSpan:` (page 2864)
Returns an initialized text table block.

Getting the Block's Enclosing Table

`- table` (page 2866)
Returns the table containing this text table block.

Getting Information About the Block's Position in Its Enclosing Table

- `startingRow` (page 2865)
  Returns the table row at which this text table block starts.
- `rowSpan` (page 2865)
  Returns the number of table rows spanned by this text table block.
- `startingColumn` (page 2865)
  Returns the table column at which this text table block starts.
- `columnSpan` (page 2864)
  Returns the number of table columns spanned by this text table block.

Instance Methods

columnSpan

Returns the number of table columns spanned by this text table block.

```
- (NSInteger)columnSpan
```

Return Value
The number of table columns spanned by this text table block.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTextTable.h`

initWithTable:startingRow:rowSpan:startingColumn:columnSpan:

Returns an initialized text table block.

```
- (id)initWithTable:(NSTextTable *)table startingRow:(NSInteger)row
    rowSpan:(NSInteger)rowSpan startingColumn:(NSInteger)col
    columnSpan:(NSInteger)colSpan
```

Parameters

table
The text table containing this text table block.

row
The table row at which the text table block starts.

rowSpan
How many rows the text table block covers.

col
The table column at which the text table block starts.

colSpan
How many columns the text table block covers.

Discussion
This is the designated initializer.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
iSpend

Declared In
`NSTextTable.h`

rowSpan

Returns the number of table rows spanned by this text table block.

```
- (NSInteger)rowSpan
```

Return Value
The number of table rows spanned by this text table block.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTextTable.h`

startingColumn

Returns the table column at which this text table block starts.

```
- (NSInteger)startingColumn
```

Return Value
The table column at which this text table block starts.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTextTable.h`

startingRow

Returns the table row at which this text table block starts.

```
- (NSInteger)startingRow
```

Return Value
The table row at which this text table block starts.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTextTable.h

table

Returns the table containing this text table block.

```
- (NSTextTable *)table
```

Return Value
The table containing this text table block.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
iSpend

Declared In
NSTextTable.h

NSTextView Class Reference

Inherits from	NSText : NSView : NSResponder : NSObject
Conforms to	NSTextInput NSUserInterfaceValidations NSTextInputClient NSChangeSpelling (NSText) NSIgnoreMisspelledWords (NSText) NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTextView.h
Companion guides	Text System Overview Text System User Interface Layer Programming Guide
Related sample code	Quartz Composer WWDC 2005 TextEdit Sketch-112 TextSizingExample TipWrapper XMLBrowser

Overview

`NSTextView` is the front-end class to the Application Kit's text system. It draws the text managed by the back-end components and handles user events to select and modify its text. `NSTextView` is the principal means to obtain a text object that caters to almost all needs for displaying and managing text at the user interface level. While `NSTextView` is a subclass of `NSText`—which declares the most general Cocoa interface to the text system—`NSTextView` adds major features beyond the capabilities of `NSText`.

About Delegate Methods

`NSTextView` communicates with its delegate through methods declared both by the `NSTextViewDelegate Protocol` protocol and by its superclass's protocol, `NSTextDelegate Protocol`. Note that all delegation messages come from the first text view.

Tasks

Initializing

- `initWithFrame:textContainer:` (page 2896)
  Initializes a text view.
- `initWithFrame:` (page 2895)
  Initializes a text view.

Registering Services Information

+ `registerForServices` (page 2879)
  Registers send and return types for the Services facility.

Accessing Text System Objects

- `setTextContainer:` (page 2945)
  Sets the receiver's text container.
- `replaceTextContainer:` (page 2916)
  Replaces the text container for the group of text system objects containing the receiver, keeping the association between the receiver and its layout manager intact.
- `textContainer` (page 2955)
  Returns the receiver's text container.
- `setTextContainerInset:` (page 2946)
  Sets the empty space the receiver leaves around its associated text container.
- `textContainerInset` (page 2956)
  Returns the empty space the receiver leaves around its text container.
- `textContainerOrigin` (page 2956)
  Returns the origin of the receiver's text container.
- `invalidateTextContainerOrigin` (page 2898)
  Invalidates the calculated origin of the text container.
- `layoutManager` (page 2904)
  Returns the layout manager that lays out text for the receiver's text container.
- `textStorage` (page 2957)
  Returns the receiver's text storage object.

Setting Graphics Attributes

- `setBackgroundColor:` (page 2930)
  Sets the receiver's background color.
- `backgroundColor` (page 2882)
  Returns the receiver's background color.

- `setDrawsBackground:` (page 2934)
  Sets whether the receiver draws its background.
- `drawsBackground` (page 2893)
  Returns whether the receiver draws its background
- `setAllowsDocumentBackgroundColorChange:` (page 2926)
  Sets whether the receiver allows its background color to change.
- `allowsDocumentBackgroundColorChange` (page 2881)
  Returns whether the receiver allows its background color to change.
- `changeDocumentBackgroundColor:` (page 2884)
  An action method used to set the background color.

Controlling Display

- `setNeedsDisplayInRect:avoidAdditionalLayout:` (page 2938)
  Marks the receiver as requiring display.
- `shouldDrawInsertionPoint` (page 2950)
  Returns whether the receiver should draw its insertion point.
- `drawInsertionPointInRect:color:turnedOn:` (page 2892)
  Draws or erases the insertion point.
- `drawViewBackgroundInRect:` (page 2893)
  Draws the background of the text view.
- `setConstrainedFrameSize:` (page 2931)
  Attempts to set the frame size as if by user action.
- `cleanUpAfterDragOperation` (page 2887)
  Releases the drag information still existing after the dragging session has completed.
- `showFindIndicatorForRange:` (page 2951)
  Causes a temporary highlighting effect to appear around the visible portion (or portions) of the specified range.

Inserting Text

- `insertText:` (page 2898)
  Inserts *`aString`* into the receiver's text at the insertion point if there is one, otherwise replacing the selection.
- `allowedInputSourceLocales` (page 2881)
  Returns an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.
- `setAllowedInputSourceLocales:` (page 2926)
  Sets an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.

Setting Behavioral Attributes

- `allowsUndo` (page 2882)
  Returns whether the receiver allows undo.
- `setAllowsUndo:` (page 2927)
  Sets whether undo support is enabled.
- `setEditable:` (page 2934)
  Controls whether the text views sharing the receiver's layout manager allow the user to edit text.
- `isEditable` (page 2902)
  Returns whether the text views sharing the receiver's layout manager allow the user to edit text.
- `setSelectable:` (page 2940)
  Controls whether the text views sharing the receiver's layout manager allow the user to select text.
- `isSelectable` (page 2904)
  Returns whether the text views sharing the receiver's layout manager allow the user to select text.
- `setFieldEditor:` (page 2935)
  Controls whether the text views sharing the receiver's layout manager behave as field editors.
- `isFieldEditor` (page 2902)
  Returns whether the text views sharing the receiver's layout manager behave as field editors.
- `setRichText:` (page 2939)
  Controls whether the text views sharing the receiver's layout manager allow the user to apply attributes to specific ranges of text.
- `isRichText` (page 2903)
  Returns whether the text views sharing the receiver's layout manager allow the user to apply attributes to specific ranges of text.
- `setImportsGraphics:` (page 2936)
  Controls whether the text views sharing the receiver's layout manager allow the user to import files by dragging.
- `importsGraphics` (page 2895)
  Returns whether the text views sharing the receiver's layout manager allow the user to import files by dragging.
- `setBaseWritingDirection:range:` (page 2931)
  Sets the base writing direction of a range of text.
- `setDefaultParagraphStyle:` (page 2932)
  Sets the receiver's default paragraph style.
- `defaultParagraphStyle` (page 2889)
  Returns the receiver's default paragraph style.
- `outline:` (page 2908)
  Adds the outline attribute to the selected text attributes if absent; removes the attribute if present.
- `allowsImageEditing` (page 2881)
  Indicates whether image attachments should permit editing of their images.
- `setAllowsImageEditing:` (page 2927)
  Specifies whether image attachments should permit editing of their images.
- `setAutomaticQuoteSubstitutionEnabled:` (page 2929)
  Enables and disables automatic quotation mark substitution.

- `isAutomaticQuoteSubstitutionEnabled` (page 2900)
  Indicates whether automatic quotation mark substitution is enabled.
- `toggleAutomaticQuoteSubstitution:` (page 2959)
  Changes the state of automatic quotation mark substitution from enabled to disabled and vice versa.
- `setAutomaticLinkDetectionEnabled:` (page 2929)
  Enables or disables automatic link detection.
- `isAutomaticLinkDetectionEnabled` (page 2899)
  Indicates whether automatic link detection is enabled.
- `toggleAutomaticLinkDetection:` (page 2958)
  Changes the state of automatic link detection from enabled to disabled and vice versa.
- `displaysLinkToolTips` (page 2890)
  Indicates whether the text view automatically supplies the destination of a link as a tooltip for text that has a link attribute.
- `setDisplaysLinkToolTips:` (page 2933)
  Enables or disables automatic display of link tooltips.
- `toggleBaseWritingDirection:` (page 2960) Deprecated in Mac OS X v10.6
  Changes the base writing direction of a paragraph between left-to-right and right-to-left.

Using the Ruler

- `setUsesRuler:` (page 2948)
  Controls whether the text views sharing the receiver's layout manager use a ruler.
- `usesRuler` (page 2966)
  Returns whether the text views sharing the receiver's layout manager use a ruler.
- `setRulerVisible:` (page 2939)
  Controls whether the scroll view enclosing text views sharing the receiver's layout manager displays the ruler.
- `isRulerVisible` (page 2904)
  Returns whether the scroll view enclosing the text views sharing the receiver's layout manager shows its ruler.

Managing the Selection

- `selectedRanges` (page 2923)
  Returns an array containing the ranges of characters selected in the receiver's layout manager.
- `setSelectedRange:` (page 2940)
  Sets the selection to the characters in a single range.
- `setSelectedRanges:` (page 2942)
  Sets the selection to the characters in an array of ranges.
- `setSelectedRange:affinity:stillSelecting:` (page 2941)
  Sets the selection to a range of characters in response to user action.
- `setSelectedRanges:affinity:stillSelecting:` (page 2943)
  Sets the selection to the characters in an array of ranges in response to user action.

- `selectionAffinity` (page 2923)
  Returns the preferred direction of selection.
- `setSelectionGranularity:` (page 2944)
  Sets the selection granularity for subsequent extension of a selection.
- `selectionGranularity` (page 2924)
  Returns the current selection granularity, used during mouse tracking to modify the range of the selection.
- `setInsertionPointColor:` (page 2937)
  Sets the color of the insertion point
- `insertionPointColor` (page 2897)
  Returns the color used to draw the insertion point.
- `updateInsertionPointStateAndRestartTimer:` (page 2964)
  Updates the insertion point's location and optionally restarts the blinking cursor timer.
- `setSelectedTextAttributes:` (page 2943)
  Sets the attributes used to indicate the selection.
- `selectedTextAttributes` (page 2923)
  Returns the attributes used to indicate the selection.
- `setMarkedTextAttributes:` (page 2938)
  Sets the attributes used to draw marked text.
- `markedTextAttributes` (page 2906)
  Returns the attributes used to draw marked text.
- `setLinkTextAttributes:` (page 2937)
  Sets the attributes used to draw the onscreen presentation of link text.
- `linkTextAttributes` (page 2905)
  Returns the attributes used to draw the onscreen presentation of link text.
- `characterIndexForInsertionAtPoint:` (page 2885)
  Returns a character index appropriate for placing a zero-length selection for an insertion point associated with the mouse at the given point.

Managing the Pasteboard

- `preferredPasteboardTypeFromArray:restrictedToTypesFromArray:` (page 2910)
  Returns whatever type on the pasteboard would be most preferred for copying data.
- `readSelectionFromPasteboard:` (page 2915)
  Reads the text view's preferred type of data from the specified pasteboard.
- `readSelectionFromPasteboard:type:` (page 2916)
  Reads data of the given type from the specified pasteboard.
- `readablePasteboardTypes` (page 2915)
  Returns the types this text view can read immediately from the pasteboard.
- `writablePasteboardTypes` (page 2968)
  Returns the pasteboard types that can be provided from the current selection.
- `writeSelectionToPasteboard:type:` (page 2968)
  Writes the current selection to the specified pasteboard using the given type.

- `writeSelectionToPasteboard:types:` (page 2969)
  Writes the current selection to the specified pasteboard under each given type.
- `validRequestorForSendType:returnType:` (page 2967)
  Returns `self` if the text view can provide and accept the specified data types, or `nil` if it can't.

Setting Text Attributes

- `alignJustified:` (page 2880)
  Applies full justification to selected paragraphs (or all text, if the receiver is a plain text object).
- `changeAttributes:` (page 2883)
  Changes the attributes of the current selection.
- `changeColor:` (page 2884)
  Sets the color of the selected text.
- `setAlignment:range:` (page 2925)
  Sets the alignment of the paragraphs containing characters in the specified range.
- `setTypingAttributes:` (page 2947)
  Sets the receiver's typing attributes.
- `typingAttributes` (page 2963)
  Returns the current typing attributes.
- `useStandardKerning:` (page 2966)
  Set the receiver to use pair kerning data for the glyphs in its selection, or for all glyphs if the receiver is a plain text view.
- `lowerBaseline:` (page 2906)
  Lowers the baseline offset of selected text by 1 point, or of all text if the receiver is a plain text view.
- `raiseBaseline:` (page 2911)
  Raises the baseline offset of selected text by 1 point, or of all text if the receiver is a plain text view.
- `turnOffKerning:` (page 2962)
  Sets the receiver to use nominal glyph spacing for the glyphs in its selection, or for all glyphs if the receiver is a plain text view.
- `loosenKerning:` (page 2905)
  Increases the space between glyphs in the receiver's selection, or in all text if the receiver is a plain text view.
- `tightenKerning:` (page 2957)
  Decreases the space between glyphs in the receiver's selection, or for all glyphs if the receiver is a plain text view.
- `useStandardLigatures:` (page 2967)
  Sets the receiver to use the standard ligatures available for the fonts and languages used when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.
- `turnOffLigatures:` (page 2962)
  Sets the receiver to use only required ligatures when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.
- `useAllLigatures:` (page 2965)
  Sets the receiver to use all ligatures available for the fonts and languages used when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.

- `toggleTraditionalCharacterShape:` (page 2962)
  Toggles the `NSCharacterShapeAttributeName` attribute at the current selection.

Clicking and Pasting

- `clickedOnLink:atIndex:` (page 2887)
  Causes the text view to act as if the user clicked on some text with the given link as the value of a link attribute associated with the text.
- `pasteAsPlainText:` (page 2909)
  Inserts the contents of the pasteboard into the receiver's text as plain text.
- `pasteAsRichText:` (page 2909)
  This action method inserts the contents of the pasteboard into the receiver's text as rich text, maintaining its attributes.

Undo Support

- `breakUndoCoalescing` (page 2883)
  Informs the receiver that it should begin coalescing successive typing operations in a new undo grouping.
- `isCoalescingUndo` (page 2901)
  Returns whether undo coalescing is in progress.

Methods for Subclasses to Use or Override

- `updateFontPanel` (page 2964)
  Updates the Font panel to contain the font attributes of the selection.
- `updateRuler` (page 2965)
  Updates the ruler view in the receiver's enclosing scroll view to reflect the selection's paragraph and marker attributes.
- `acceptableDragTypes` (page 2879)
  Returns the data types that the receiver accepts as the destination view of a dragging operation.
- `updateDragTypeRegistration` (page 2963)
  Updates the acceptable drag types of all text views associated with the receiver's layout manager.
- `selectionRangeForProposedRange:granularity:` (page 2924)
  Returns an adjusted selected range based on the selection granularity.
- `rangeForUserCharacterAttributeChange` (page 2911)
  Returns the range of characters affected by an action method that changes character (not paragraph) attributes.
- `rangesForUserCharacterAttributeChange` (page 2913)
  Returns an array containing the ranges of characters affected by an action method that changes character (not paragraph) attributes.
- `rangeForUserParagraphAttributeChange` (page 2912)
  Returns the range of characters affected by an action method that changes paragraph (not character) attributes.

- `rangesForUserParagraphAttributeChange` (page 2914)
  Returns an array containing the ranges of characters affected by a method that changes paragraph (not character) attributes.
- `rangeForUserTextChange` (page 2913)
  Returns the range of characters affected by a method that changes characters (as opposed to attributes).
- `rangesForUserTextChange` (page 2914)
  Returns an array containing the ranges of characters affected by a method that changes characters (as opposed to attributes).
- `shouldChangeTextInRange:replacementString:` (page 2949)
  Initiates a series of delegate messages (and general notifications) to determine whether modifications can be made to the characters and attributes of the receiver's text.
- `shouldChangeTextInRanges:replacementStrings:` (page 2950)
  Initiates a series of delegate messages (and general notifications) to determine whether modifications can be made to the characters and attributes of the receiver's text.
- `didChangeText` (page 2890)
  Sends out necessary notifications when a text change completes.
- `setSmartInsertDeleteEnabled:` (page 2944)
  Controls whether the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation.
- `smartInsertDeleteEnabled` (page 2953)
  Returns whether the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation.
- `smartDeleteRangeForProposedRange:` (page 2951)
  Returns an extended range that includes adjacent whitespace that should be deleted along with the proposed range in order to preserve proper spacing and punctuation.
- `smartInsertAfterStringForString:replacingRange:` (page 2952)
  Returns any whitespace that needs to be added after the string to preserve proper spacing and punctuation when the string replaces the characters in the specified range.
- `smartInsertBeforeStringForString:replacingRange:` (page 2952)
  Returns any whitespace that needs to be added before the string to preserve proper spacing and punctuation when the string replaces the characters in the specified range.
- `smartInsertForString:replacingRange:beforeString:afterString:` (page 2953)
  Determines whether whitespace needs to be added around the string to preserve proper spacing and punctuation when it replaces the characters in the specified range.
- `toggleSmartInsertDelete:` (page 2961)
  Changes the state of smart insert and delete from enabled to disabled and vice versa.

Working With the Spelling Checker

- `isContinuousSpellCheckingEnabled` (page 2902)
  Indicates whether the receiver has continuous spell checking enabled.
- `setContinuousSpellCheckingEnabled:` (page 2932)
  Enables or disables continuous spell checking.
- `spellCheckerDocumentTag` (page 2954)
  Returns a tag identifying the text view's text as a document for the spell checker server.

- `toggleContinuousSpellChecking:` (page 2960)
  Toggles whether continuous spell checking is enabled for the receiver.
- `setGrammarCheckingEnabled:` (page 2936)
  Enables and disables grammar checking.
- `isGrammarCheckingEnabled` (page 2903)
  Indicates whether or not grammar checking is enabled.
- `toggleGrammarChecking:` (page 2961)
  Changes the state of grammar checking from enabled to disabled and vice versa.
- `setSpellingState:range:` (page 2945)
  Sets the spelling state, which controls the display of the spelling and grammar indicators on the given text range.

NSRulerView Client Methods

- `rulerView:didMoveMarker:` (page 2918)
  Modifies the paragraph style of the paragraphs containing the selection to record the new location of the marker.
- `rulerView:willMoveMarker:toLocation:` (page 2922)
  Returns a potentially modified location to which the marker should be moved.
- `rulerView:shouldMoveMarker:` (page 2920)
  Returns whether the marker should be moved.
- `rulerView:didRemoveMarker:` (page 2919)
  Modifies the paragraph style of the paragraphs containing the selection—if possible—by removing the specified marker.
- `rulerView:shouldRemoveMarker:` (page 2921)
  Returns whether the marker should be removed.
- `rulerView:didAddMarker:` (page 2917)
  Modifies the paragraph style of the paragraphs containing the selection to accommodate a new marker.
- `rulerView:shouldAddMarker:` (page 2920)
  Returns whether a new marker can be added.
- `rulerView:willAddMarker:atLocation:` (page 2921)
  Returns a potentially modified location to which the marker should be added.
- `rulerView:handleMouseDown:` (page 2919)
  Adds a left tab marker to the ruler at the location clicked.

Assigning a Delegate

- `setDelegate:` (page 2933)
  Sets the delegate for all text views sharing the receiver's layout manager.
- `delegate` (page 2889)
  Returns the delegate used by the receiver and all other text views sharing the receiver's layout manager.

Dragging

- `dragImageForSelectionWithEvent:origin:` (page 2890)
  Returns an appropriate drag image for the drag initiated by the specified event.
- `dragOperationForDraggingInfo:type:` (page 2891)
  Returns the type of drag operation that should be performed if the image were released now.
- `dragSelectionWithEvent:offset:slideBack:` (page 2892)
  Begins dragging the current selected text range.
- `acceptsGlyphInfo` (page 2880)
  Returns whether the receiver accepts the glyph info attribute.
- `setAcceptsGlyphInfo:` (page 2925)
  Sets whether the receiver accepts the glyph info attribute.

Speaking Text

- `startSpeaking:` (page 2954)
  Speaks the selected text, or all text if no selection.
- `stopSpeaking:` (page 2955)
  Stops the speaking of text.

Working with Panels

- `setUsesFontPanel:` (page 2947)
  Controls whether the text views sharing the receiver's layout manager use the Font panel and Font menu.
- `usesFontPanel` (page 2966)
  Returns whether the text views sharing the receiver's layout manager use the Font panel.
- `setUsesFindPanel:` (page 2947)
  Specifies whether the receiver allows for a find panel.
- `usesFindPanel` (page 2965)
  Returns whether the receiver allows for a find panel.
- `performFindPanelAction:` (page 2910)
  Performs a find panel action specified by the sender's tag.
- `orderFrontLinkPanel:` (page 2907)
  Brings forward a panel allowing the user to manipulate links in the text view.
- `orderFrontListPanel:` (page 2907)
  Brings forward a panel allowing the user to manipulate text lists in the text view.
- `orderFrontSpacingPanel:` (page 2907)
  Brings forward a panel allowing the user to manipulate text line heights, interline spacing, and paragraph spacing, in the text view.
- `orderFrontTablePanel:` (page 2908)
  Brings forward a panel allowing the user to manipulate text tables in the text view.
- `orderFrontSubstitutionsPanel:` (page 2908)
  Brings forward a panel allowing the user to specify string substitutions in the text view.

Text Completion

- `complete:` (page 2888)
  Invokes completion in a text view.
- `completionsForPartialWordRange:indexOfSelectedItem:` (page 2888)
  Returns an array of potential completions, in the order to be presented, representing possible word completions available from a partial word.
- `insertCompletion:forPartialWordRange:movement:isFinal:` (page 2897)
  Inserts the selected completion into the text at the appropriate location.
- `rangeForUserCompletion` (page 2912)
  Returns the partial range from the most recent beginning of a word up to the insertion point.

Text Checking and Substitutions

- `checkTextInDocument:` (page 2885)
  Performs the default text checking on the entire document.
- `checkTextInSelection:` (page 2886)
  Performs the default text checking on the current selection.
- `checkTextInRange:types:options:` (page 2886)
  Check and replace the text in the range using the specified checking types and options.
- `handleTextCheckingResults:forRange:types:options:orthography:wordCount:` (page 2894)
  Handles the text checking results returned by the text view
- `enabledTextCheckingTypes` (page 2894)
  Returns the default text checking types.
- `setEnabledTextCheckingTypes:` (page 2935)
  Sets the default text checking types.
- `isAutomaticDashSubstitutionEnabled` (page 2899)
  Returns whether automatic dash substitution is enabled.
- `setAutomaticDashSubstitutionEnabled:` (page 2928)
  Sets whether automatic dash substitution is enabled.
- `toggleAutomaticDashSubstitution:` (page 2957)
  Toggles the state of the automatic dash substitution.
- `isAutomaticDataDetectionEnabled` (page 2899)
  Returns whether automatic data detection is enabled.
- `setAutomaticDataDetectionEnabled:` (page 2928)
  Sets whether automatic data detection is enabled.
- `toggleAutomaticDataDetection:` (page 2958)
  Toggles the state of the automatic data detection.
- `isAutomaticSpellingCorrectionEnabled` (page 2900)
  Returns whether automatic spelling correction is enabled.
- `setAutomaticSpellingCorrectionEnabled:` (page 2929)
  Sets whether automatic spelling correction is enabled.
- `toggleAutomaticSpellingCorrection:` (page 2959)
  Toggles the state of the automatic spelling correction.

- `isAutomaticTextReplacementEnabled` (page 2901)
  Returns whether automatic text replacement is enabled.
- `setAutomaticTextReplacementEnabled:` (page 2930)
  Sets whether automatic text replacement is enabled.
- `toggleAutomaticTextReplacement:` (page 2960)
  Toggles the state of the automatic text replacement.

Changing First Responder Status

- `becomeFirstResponder` (page 2883) Available in Mac OS X v10.0 through Mac OS X v10.4
  Informs the receiver that it's becoming the first responder. (Deprecated. Use the `NSWindow` method `makeFirstResponder:` (page 3344) to make a text view the first responder.)
- `resignFirstResponder` (page 2917) Available in Mac OS X v10.0 through Mac OS X v10.4
  Notifies the receiver that it's been asked to relinquish its status as first responder. (Deprecated. Use the `NSWindow` method `makeFirstResponder:` (page 3344) to make a text view the first responder.)

Class Methods

registerForServices

Registers send and return types for the Services facility.

```
+ (void)registerForServices
```

Discussion
This method is invoked automatically when the first instance of a text view is created; you should never need to invoke it directly.

Subclasses of `NSTextView` that wish to add support for new service types should override `registerForServices` to call `super` and then register their own new types.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextView.h`

Instance Methods

acceptableDragTypes

Returns the data types that the receiver accepts as the destination view of a dragging operation.

```
- (NSArray *)acceptableDragTypes
```

Return Value
The data types that the receiver accepts as the destination view of a dragging operation.

Discussion
These types are automatically registered as necessary by the text view. Subclasses should override this method as necessary to add their own types to those returned by `NSTextView`'s implementation. They must then also override the appropriate methods of the `NSDraggingDestination` protocol to support import of those types. See that protocol's specification for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- updateDragTypeRegistration` (page 2963)

Declared In
`NSTextView.h`

acceptsGlyphInfo

Returns whether the receiver accepts the glyph info attribute.

```
- (BOOL)acceptsGlyphInfo
```

Return Value
`YES` if the receiver accepts the `NSGlyphInfoAttributeName` attribute from text input sources such as input methods and the pasteboard, `NO` otherwise.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- setAcceptsGlyphInfo:` (page 2925)

Declared In
`NSTextView.h`

alignJustified:

Applies full justification to selected paragraphs (or all text, if the receiver is a plain text object).

```
- (void)alignJustified:(id)sender
```

Parameters
sender
The control that sent the message; may be `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- alignCenter:` (page 2718) (NSText)
`- alignLeft:` (page 2718) (NSText)

- alignRight: (page 2719) (NSText)
- alignment (page 2718) (NSText)
- setAlignment: (page 2733) (NSText)

Declared In
NSTextView.h

allowedInputSourceLocales

Returns an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.

```
- (NSArray *)allowedInputSourceLocales
```

Return Value
The locale identifiers of allowed input sources.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setAllowedInputSourceLocales: (page 2926)

Declared In
NSTextView.h

allowsDocumentBackgroundColorChange

Returns whether the receiver allows its background color to change.

```
- (BOOL)allowsDocumentBackgroundColorChange
```

Return Value
YES if the receiver allows its background color to change, otherwise NO.

Discussion
This corresponds to the background color of the entirety of the text view, not just to a selected range of text.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setAllowsDocumentBackgroundColorChange: (page 2926)
- changeDocumentBackgroundColor: (page 2884)

Declared In
NSTextView.h

allowsImageEditing

Indicates whether image attachments should permit editing of their images.

```
- (BOOL)allowsImageEditing
```

Return Value
`YES` if image editing is allowed; otherwise, `NO`.

Discussion
For image editing to be allowed, the text view must be editable and the text attachment cell must support image editing.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setAllowsImageEditing:` (page 2927)

Declared In
`NSTextView.h`

allowsUndo

Returns whether the receiver allows undo.

```
- (BOOL)allowsUndo
```

Return Value
`YES` if the receiver allows undo, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAllowsUndo:` (page 2927)

Declared In
`NSTextView.h`

backgroundColor

Returns the receiver's background color.

```
- (NSColor *)backgroundColor
```

Return Value
The receiver's background color.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- drawsBackground` (page 2893)
`- setBackgroundColor:` (page 2930)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSTextView.h

becomeFirstResponder

Informs the receiver that it's becoming the first responder. (Available in Mac OS X v10.0 through Mac OS X v10.4. Use the NSWindow method makeFirstResponder: (page 3344) to make a text view the first responder.)

```
- (BOOL)becomeFirstResponder
```

Return Value
Returns YES.

Discussion
If the previous first responder was not an *NSTextView* on the same NSLayoutManager as the receiving NSTextView, this method draws the selection and updates the insertion point if necessary.

Use the NSWindow method makeFirstResponder: (page 3344), not this method, to make a text view the first responder. Never invoke this method directly.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.4.

See Also
- resignFirstResponder (page 2917)

Declared In
NSTextView.h

breakUndoCoalescing

Informs the receiver that it should begin coalescing successive typing operations in a new undo grouping.

```
- (void)breakUndoCoalescing
```

Special Considerations

This method should be invoked when saving the receiver's contents to preserve proper tracking of unsaved changes and the document's dirty state.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSTextView.h

changeAttributes:

Changes the attributes of the current selection.

```
- (void)changeAttributes:(id)sender
```

Parameters

sender

The control that sent the message. Must respond to `convertAttributes:`.

Discussion

This method changes the attributes by invoking `convertAttributes:` (page 1220) on *sender* and applying the returned attributes to the appropriate text. See the `NSFontManager` class reference for more information on attribute conversion.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`NSTextView.h`

changeColor:

Sets the color of the selected text.

```
- (void)changeColor:(id)sender
```

Parameters

sender

The control that sent the message. `NSTextView`'s implementation sends a `color` (page 731) message to *sender* to get the new color.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextView.h`

changeDocumentBackgroundColor:

An action method used to set the background color.

```
- (void)changeDocumentBackgroundColor:(id)sender
```

Parameters

sender

The control that wants to set the background color.

Discussion

This method gets the new color by sending a `color` (page 731) message to *sender*.

This will only set the background color if `allowsDocumentBackgroundColorChange` (page 2881)returns `YES`.

Availability

Available in Mac OS X v10.3 and later.

See Also
- `setAllowsDocumentBackgroundColorChange:` (page 2926)
- `allowsDocumentBackgroundColorChange` (page 2881)

Declared In
`NSTextView.h`

characterIndexForInsertionAtPoint:

Returns a character index appropriate for placing a zero-length selection for an insertion point associated with the mouse at the given point.

```
- (NSUInteger)characterIndexForInsertionAtPoint:(NSPoint)point
```

Parameters

point

The point for which to return an index, in view coordinates.

Return Value
The character index for the insertion point.

Discussion
This method should be used for insertion points associated with mouse clicks, drag events, and so forth. For other purposes, it is better to use `NSLayoutManager` methods.

The `NSTextInput` method `characterIndexForPoint:` (page 3863) is not suitable for this role; it is intended only for uses related to text input methods.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTextView.h`

checkTextInDocument:

Performs the default text checking on the entire document.

```
- (void)checkTextInDocument:(id)sender
```

Parameters

sender

The control sending the message. May be `nil`.

Discussion
Immediately performs the text checking and replaces the document content with the checked content.

The checks performed are specified by `setEnabledTextCheckingTypes:` (page 2935);

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setEnabledTextCheckingTypes:` (page 2935)

- checkTextInSelection: (page 2886)

Declared In
NSTextView.h

checkTextInRange:types:options:

Check and replace the text in the range using the specified checking types and options.

```
- (void)checkTextInRange:(NSRange)range types:(NSTextCheckingTypes)checkingTypes
    options:(NSDictionary *)options
```

Parameters

range

The range to check.

checkingTypes

The type of checking to be performed, passed by-reference. The possible constants are listed in NSTextCheckingTypes and can be combined using the C bit-wise OR operator to perform multiple checks at the same time.

options

A dictionary of values used during the checking process to perform. See Spell Checking Option Dictionary Keys (page 2537) for the supported values.

Discussion
This method calls the delegate method textView:willCheckTextInRange:options:types: (page 3893) allowing you to modify the parameters before the checking occurs.

This method usually would not be called directly, since NSTextView itself will call it as needed, but it can be overridden.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSTextView.h

checkTextInSelection:

Performs the default text checking on the current selection.

```
- (void)checkTextInSelection:(id)sender
```

Parameters

sender

The control sending the message. May be nil.

Discussion
Immediately performs the text checking and replaces the selection with the checked content.

The checks performed are specified by setEnabledTextCheckingTypes: (page 2935);

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setEnabledTextCheckingTypes:` (page 2935)
- `checkTextInDocument:` (page 2885)

Declared In
`NSTextView.h`

cleanUpAfterDragOperation

Releases the drag information still existing after the dragging session has completed.

```
- (void)cleanUpAfterDragOperation
```

Discussion
Subclasses may override this method to clean up any additional data structures used for dragging. In your overridden method, be sure to invoke `super`'s implementation of this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextView.h`

clickedOnLink:atIndex:

Causes the text view to act as if the user clicked on some text with the given link as the value of a link attribute associated with the text.

```
- (void)clickedOnLink:(id)link atIndex:(NSUInteger)charIndex
```

Parameters

link
The link that was clicked; the value of `NSLinkAttributeName` (page 272).

charIndex
The character index where the click occurred, indexed within the text storage.

Discussion
If, for instance, you have a special attachment cell that can follow links, you can use this method to ask the text view to follow a link once you decide it should. In addition, this method is invoked by the text view during mouse tracking if the user clicks a link.

The *charIndex* parameter is a character index somewhere in the range of the link attribute. If the user actually physically clicked the link, then it should be the character that was originally clicked. In some cases a link may be opened indirectly or programmatically, in which case a character index somewhere in the range of the link attribute is supplied.

This method sends the `textView:clickedOnLink:atIndex:` (page 3883) delegate message if the delegate implements it, so that the delegate can handle the click.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textView:clickedOnLink:atIndex:` (page 3883) (NSTextViewDelegate)

Declared In
`NSTextView.h`

complete:

Invokes completion in a text view.

```
- (void)complete:(id)sender
```

Parameters

sender
The control sending the message. May be `nil`.

Discussion
By default invoked using the F5 key, this method provides users with a choice of completions for the word currently being typed. May be invoked programmatically if autocompletion is desired by a client of the text system. You can change the key invoking this method using the text system's key bindings mechanism; see "Text System Defaults and Key Bindings" for an explanation of the procedure.

The delegate may replace or modify the list of possible completions by implementing `textView:completions:forPartialWordRange:indexOfSelectedItem:` (page 3884). Subclasses may control the list by overriding `completionsForPartialWordRange:indexOfSelectedItem:` (page 2888).

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
SearchField

Declared In
`NSTextView.h`

completionsForPartialWordRange:indexOfSelectedItem:

Returns an array of potential completions, in the order to be presented, representing possible word completions available from a partial word.

```
- (NSArray *)completionsForPartialWordRange:(NSRange)charRange
    indexOfSelectedItem:(NSInteger *)index
```

Parameters

charRange
The range of characters of the partial word to be completed.

index
On return, optionally set to the completion that should be initially selected. The default is 0, and –1 indicates no selection.

Return Value
An array of potential completions, in the order to be presented, representing possible word completions available from a partial word at *charRange*. Returning `nil` or a zero-length array suppresses completion.

Discussion
May be overridden by subclasses to modify or override the list of possible completions.

This method should call the delegate method `textView:completions:forPartialWordRange:indexOfSelectedItem:` (page 3884)s if the delegate implements such a method.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSTextView.h`

defaultParagraphStyle

Returns the receiver's default paragraph style.

```
- (NSParagraphStyle *)defaultParagraphStyle
```

Return Value
The receiver's default paragraph style.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setDefaultParagraphStyle:` (page 2932)

Declared In
`NSTextView.h`

delegate

Returns the delegate used by the receiver and all other text views sharing the receiver's layout manager.

```
- (id < NSTextViewDelegate >)delegate
```

Return Value
The delegate used by the receiver and all other text views sharing the receiver's layout manager, or `nil` if there is none.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setDelegate:` (page 2933)

Related Sample Code
TextLinks

Declared In
`NSTextView.h`

didChangeText

Sends out necessary notifications when a text change completes.

```
- (void)didChangeText
```

Discussion
Invoked automatically at the end of a series of changes, this method posts an `NSTextDidChangeNotification` (page 2752) to the default notification center, which also results in the delegate receiving an `NSText` delegate `textDidChange:` message.

Subclasses implementing methods that change their text should invoke this method at the end of those methods. See Subclassing NSTextView for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- shouldChangeTextInRange:replacementString:` (page 2949)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSTextView.h`

displaysLinkToolTips

Indicates whether the text view automatically supplies the destination of a link as a tooltip for text that has a link attribute.

```
- (BOOL)displaysLinkToolTips
```

Return Value
`YES` if link tooltips are automatically displayed; otherwise, `NO`.

Discussion
The default value for this feature is `YES`; clients who do not wish tooltips to be displayed automatically must explicitly disable it.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setDisplaysLinkToolTips:` (page 2933)

Declared In
`NSTextView.h`

dragImageForSelectionWithEvent:origin:

Returns an appropriate drag image for the drag initiated by the specified event.

```
- (NSImage *)dragImageForSelectionWithEvent:(NSEvent *)event
    origin:(NSPointPointer)origin
```

Parameters

event

The event that initiated the drag session.

origin

On return, the lower-left point of the image in view coordinates.

Return Value

An appropriate drag image for the drag initiated by *event*. May be `nil`, in which case a default icon will be used.

Discussion

This method is used by `dragSelectionWithEvent:offset:slideBack:` (page 2892). It can be called by others who need such an image, or can be overridden by subclasses to return a different image.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextView.h`

dragOperationForDraggingInfo:type:

Returns the type of drag operation that should be performed if the image were released now.

```
- (NSDragOperation)dragOperationForDraggingInfo:(id < NSDraggingInfo >)dragInfo
    type:(NSString *)type
```

Parameters

dragInfo

The drag information.

type

The pasteboard type that will be read from the dragging pasteboard.

Return Value

The drag operation that should be performed if the image were released now.

Discussion

The returned value should be one of the following:

Option	Meaning
`NSDragOperationCopy`	The data represented by the image will be copied.
`NSDragOperationLink`	The data will be shared.
`NSDragOperationGeneric`	The operation will be defined by the destination.
`NSDragOperationPrivate`	The operation is negotiated privately between the source and the destination.

If none of the operations is appropriate, this method should return `NSDragOperationNone`.

This method is called repeatedly from `draggingEntered:` (page 3655) and `draggingUpdated:` (page 3656) as the user drags the image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `draggingEntered:` (page 3655) (NSDraggingDestination)
- `draggingUpdated:` (page 3656) (NSDraggingDestination)

Declared In
`NSTextView.h`

dragSelectionWithEvent:offset:slideBack:

Begins dragging the current selected text range.

```
- (BOOL)dragSelectionWithEvent:(NSEvent *)event offset:(NSSize)mouseOffset
    slideBack:(BOOL)slideBack
```

Parameters

event
The event that initiated dragging the selection.

mouseOffset
The cursor's current location relative to the mouse-down *event*.

slideBack
`YES` if the image being dragged should slide back to its original position if the drag does not succeed, `NO` otherwise.

Return Value
`YES` if the drag can be successfully initiated, `NO` otherwise.

Discussion
Primarily for subclasses, who can override it to intervene at the beginning of a drag.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextView.h`

drawInsertionPointInRect:color:turnedOn:

Draws or erases the insertion point.

```
- (void)drawInsertionPointInRect:(NSRect)aRect color:(NSColor *)aColor
    turnedOn:(BOOL)flag
```

Parameters

aRect

The rectangle in which to draw the insertion point.

aColor

The color with which to draw the insertion point.

flag

YES to draw the insertion point, NO to erase it.

Special Considerations

The focus must be locked on the receiver when this method is invoked. You should not need to invoke this method directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- insertionPointColor (page 2897)
- shouldDrawInsertionPoint (page 2950)
- backgroundColor (page 2882)
- lockFocus (page 3187) (NSView)

Declared In
NSTextView.h

drawsBackground

Returns whether the receiver draws its background

```
- (BOOL)drawsBackground
```

Return Value
YES if the receiver draws its background, NO otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- backgroundColor (page 2882)
- setDrawsBackground: (page 2934)

Declared In
NSTextView.h

drawViewBackgroundInRect:

Draws the background of the text view.

```
- (void)drawViewBackgroundInRect:(NSRect)rect
```

Parameters

rect

The rectangle in which to draw the background.

Discussion

Subclasses can override this method to perform additional drawing behind the text.

Availability

Available in Mac OS X v10.3 and later.

Declared In

NSTextView.h

enabledTextCheckingTypes

Returns the default text checking types.

```
- (NSTextCheckingTypes)enabledTextCheckingTypes
```

Return Value

The currently enabled default text checking types

Availability

Available in Mac OS X v10.6 and later.

See Also

- .: (page 2935)

Declared In

NSTextView.h

handleTextCheckingResults:forRange:types:options:orthography:wordCount:

Handles the text checking results returned by the text view

```
- (void)handleTextCheckingResults:(NSArray *)results forRange:(NSRange)range
    types:(NSTextCheckingTypes)checkingTypes options:(NSDictionary *)options
    orthography:(NSOrthography *)orthography wordCount:(NSInteger)wordCount
```

Parameters

results

An array of NSTextCheckingResult objects.

range

The range of text that was checked.

checkingTypes

The type of checking performed. The possible constants are listed in NSTextCheckingTypes and can be combined using the C bit-wise OR operator to perform multiple checks at the same time.

options

The dictionary of values used during the checking process to perform. See Spell Checking Option Dictionary Keys (page 2537) for the supported values.

orthography
The orthography of the checked text.

wordCount
The number of words.

Discussion
The `NSTextViewDelegate Protocol` offers a method, `textView:didCheckTextInRange:types:options:results:orthography:wordCount:` (page 3884) that is called after the checking is performed, allowing you to modify the results.

This method usually would not be called directly, since `NSTextView` itself will call it as needed, but it can be overridden.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTextView.h`

importsGraphics

Returns whether the text views sharing the receiver's layout manager allow the user to import files by dragging.

```
- (BOOL)importsGraphics
```

Return Value
`YES` if the user is allowed to import files by dragging onto the text views sharing the receiver's layout manager, `NO` otherwise.

Discussion
Text views that are set to accept dragged files are also set to allow rich text. By default, text views don't accept dragged files but do allow rich text.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- isRichText` (page 2903)
`- textStorage` (page 2957)
`+ attributedStringWithAttachment:` (page 252) (NSAttributedString Additions)
`insertAttributedString:atIndex:` (NSMutableAttributedString)
`- setImportsGraphics:` (page 2936)

Declared In
`NSTextView.h`

initWithFrame:

Initializes a text view.

```
- (id)initWithFrame:(NSRect)frameRect
```

Parameters

frameRect

The frame rectangle of the text view.

Return Value
An initialized text view.

Discussion
This method creates the entire collection of objects associated with a text view—its text container, layout manager, and text storage—and invokes `initWithFrame:textContainer:` (page 2896).

This method creates the text web in such a manner that the text view is the principal owner of the objects in the web.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextView.h`

initWithFrame:textContainer:

Initializes a text view.

```
- (id)initWithFrame:(NSRect)frameRect textContainer:(NSTextContainer *)aTextContainer
```

Parameters

frameRect

The frame rectangle of the text view.

aTextContainer

The text container of the text view.

Return Value
An initialized text view.

Discussion
This method is the designated initializer for `NSTextView` objects.

Unlike `initWithFrame:` (page 2895), which builds up an entire group of text-handling objects, you use this method after you've created the other components of the text-handling system—a text storage object, a layout manager, and a text container. Assembling the components in this fashion means that the text storage, not the text view, is the principal owner of the component objects.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- initWithFrame:` (page 2895)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
Sketch-112
TextLayoutDemo
TextSizingExample

TextViewConfig

Declared In
NSTextView.h

insertCompletion:forPartialWordRange:movement:isFinal:

Inserts the selected completion into the text at the appropriate location.

```
- (void)insertCompletion:(NSString *)word forPartialWordRange:(NSRange)charRange
    movement:(NSInteger)movement isFinal:(BOOL)flag
```

Parameters

word
The completion to insert.

charRange
The character range of the text being completed.

movement
The direction of movement. For possible values see the NSText Constants section. This value allows subclasses to distinguish between canceling completion and selection by arrow keys, by return, by tab, or by other means such as clicking.

flag
NO while the user navigates through the potential text completions, YES when a completion is definitively selected or cancelled and the original value is reinserted.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTextView.h

insertionPointColor

Returns the color used to draw the insertion point.

```
- (NSColor *)insertionPointColor
```

Return Value
The color used to draw the insertion point.

Availability
Available in Mac OS X v10.0 and later.

See Also
- drawInsertionPointInRect:color:turnedOn: (page 2892)
- shouldDrawInsertionPoint (page 2950)
- setInsertionPointColor: (page 2937)

Declared In
NSTextView.h

insertText:

Inserts *aString* into the receiver's text at the insertion point if there is one, otherwise replacing the selection.

```
- (void)insertText:(id)aString
```

Parameters

aString

The string to insert. *aString* can be either an `NSString` object or an `NSAttributedString` object.

Discussion

The inserted text is assigned the current typing attributes.

This method is the means by which text typed by the user enters an NSTextView. See the `NSInputManager` class and `NSTextInput` protocol specifications for more information.

This method is the entry point for inserting text typed by the user and is generally not suitable for other purposes. Programmatic modification of the text is best done by operating on the text storage directly. Because this method pertains to the actions of the user, the text view must be editable for the insertion to work.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `typingAttributes` (page 2963)

Related Sample Code

BetterAuthorizationSample
LSMSmartCategorizer
SampleScannerApp
SBSetFinderComment
TextViewDelegate

Declared In

`NSTextView.h`

invalidateTextContainerOrigin

Invalidates the calculated origin of the text container.

```
- (void)invalidateTextContainerOrigin
```

Discussion

This method is invoked automatically; you should never need to invoke it directly. Usually called because the text view has been resized or the contents of the text container have changed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `textContainer` (page 2955)
- `textContainerOrigin` (page 2956)

Declared In
NSTextView.h

isAutomaticDashSubstitutionEnabled

Returns whether automatic dash substitution is enabled.

```
- (BOOL)isAutomaticDashSubstitutionEnabled
```

Return Value
YES if it is enabled, otherwise NO.

Discussion
Turning on automatic dash substitution enables automatic conversion of sequences of ASCII hyphen (-) characters to typographic dashes.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setAutomaticDashSubstitutionEnabled: (page 2928)
- toggleAutomaticDashSubstitution: (page 2957)

Declared In
NSTextView.h

isAutomaticDataDetectionEnabled

Returns whether automatic data detection is enabled.

```
- (BOOL)isAutomaticDataDetectionEnabled
```

Return Value
YES if it is enabled, otherwise NO.

Discussion
Automatic data detection enables detection of dates, addresses, and phone numbers.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setAutomaticDataDetectionEnabled: (page 2928)
- toggleAutomaticDataDetection: (page 2958)

Declared In
NSTextView.h

isAutomaticLinkDetectionEnabled

Indicates whether automatic link detection is enabled.

```
- (BOOL)isAutomaticLinkDetectionEnabled
```

Return Value
`YES` if automatic link detection is enabled; otherwise, `NO`.

Discussion
Automatic link detection causes strings representing URLs typed in the view to be automatically made into links to those URLs.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setAutomaticLinkDetectionEnabled:` (page 2929)
`- toggleAutomaticLinkDetection:` (page 2958)
`URLAtIndex:effectiveRange:` (page 270) (`NSAttributedString`)

Declared In
`NSTextView.h`

isAutomaticQuoteSubstitutionEnabled

Indicates whether automatic quotation mark substitution is enabled.

```
- (BOOL)isAutomaticQuoteSubstitutionEnabled
```

Return Value
`YES` if automatic quotation mark substitution is enabled; otherwise, `NO`.

Discussion
Automatic quote substitution causes ASCII quotation marks and apostrophes to be automatically replaced, on a context-dependent basis, with more typographically accurate symbols.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setAutomaticQuoteSubstitutionEnabled:` (page 2929)
`- toggleAutomaticQuoteSubstitution:` (page 2959)

Declared In
`NSTextView.h`

isAutomaticSpellingCorrectionEnabled

Returns whether automatic spelling correction is enabled.

```
- (BOOL)isAutomaticSpellingCorrectionEnabled
```

Return Value
`YES` if it is enabled, otherwise `NO`.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setAutomaticSpellingCorrectionEnabled:` (page 2929)
- `toggleAutomaticSpellingCorrection:` (page 2959)

Declared In
`NSTextView.h`

isAutomaticTextReplacementEnabled

Returns whether automatic text replacement is enabled.

```
- (BOOL)isAutomaticTextReplacementEnabled
```

Return Value
`YES` if it is enabled, otherwise `NO`.

Discussion
Turning on automatic text replacement enables automatic substitution of a variety of static text items based on user preferences.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setAutomaticTextReplacementEnabled:` (page 2930)
- `toggleAutomaticTextReplacement:` (page 2960)

Declared In
`NSTextView.h`

isCoalescingUndo

Returns whether undo coalescing is in progress.

```
- (BOOL)isCoalescingUndo
```

Return Value
`YES` if undo coalescing is in progress, otherwise `NO`.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `breakUndoCoalescing` (page 2883)

Declared In
`NSTextView.h`

isContinuousSpellCheckingEnabled

Indicates whether the receiver has continuous spell checking enabled.

```
- (BOOL)isContinuousSpellCheckingEnabled
```

Return Value
`YES` if the receiver has continuous spell checking enabled, otherwise,`NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setContinuousSpellCheckingEnabled:` (page 2932)
- `toggleContinuousSpellChecking:` (page 2960)

Declared In
`NSTextView.h`

isEditable

Returns whether the text views sharing the receiver's layout manager allow the user to edit text.

```
- (BOOL)isEditable
```

Return Value
`YES` if the text views sharing the receiver's layout manager allow the user to edit text, `NO` otherwise.

Discussion
If a text view is editable, it's also selectable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isSelectable` (page 2904)
- `setEditable:` (page 2934)

Declared In
`NSTextView.h`

isFieldEditor

Returns whether the text views sharing the receiver's layout manager behave as field editors.

```
- (BOOL)isFieldEditor
```

Return Value
`YES` if the text views sharing the receiver's layout manager behave as field editors, `NO` otherwise.

Discussion
Field editors interpret Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder. Non-field editors instead accept these characters as text input. See Text Fields, Text Views, and the Field Editor for more information on field editors. By default, text views don't behave as field editors.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFieldEditor:` (page 2935)

Declared In
`NSTextView.h`

isGrammarCheckingEnabled

Indicates whether or not grammar checking is enabled.

```
- (BOOL)isGrammarCheckingEnabled
```

Return Value
`YES` if grammar checking is enabled; otherwise, `NO`.

Discussion
If grammar checking is enabled, then it is performed alongside spell checking, whenever the text view checks spelling, whether continuously or manually.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setGrammarCheckingEnabled:` (page 2936)
- `toggleGrammarChecking:` (page 2961)

Declared In
`NSTextView.h`

isRichText

Returns whether the text views sharing the receiver's layout manager allow the user to apply attributes to specific ranges of text.

```
- (BOOL)isRichText
```

Return Value
`YES` if the user is allowed to apply attributes to specific ranges of text in text views sharing the receiver's layout manager, `NO` otherwise.

Discussion
Text fields that don't allow rich text also don't accept dragged files. By default, text views let the user apply multiple attributes to text, but don't accept dragged files.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `importsGraphics` (page 2895)
- `textStorage` (page 2957)

- setRichText: (page 2939)

Declared In
NSTextView.h

isRulerVisible

Returns whether the scroll view enclosing the text views sharing the receiver's layout manager shows its ruler.

```
- (BOOL)isRulerVisible
```

Return Value
YES if the scroll view enclosing the text views sharing the receiver's layout manager shows its ruler, NO otherwise. The default is NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- usesRuler (page 2966)
- setRulerVisible: (page 2939)
- toggleRuler: (page 2745) (NSText)

Declared In
NSTextView.h

isSelectable

Returns whether the text views sharing the receiver's layout manager allow the user to select text.

```
- (BOOL)isSelectable
```

Return Value
YES if the user is allowed to select text of all text views sharing the receiver's layout manager, NO otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isEditable (page 2902)
- setSelectable: (page 2940)

Declared In
NSTextView.h

layoutManager

Returns the layout manager that lays out text for the receiver's text container.

```
- (NSLayoutManager *)layoutManager
```

Return Value
The layout manager that lays out text for the receiver's text container, or `nil` if there's no such object, such as when a text view isn't linked into a group of text objects.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textContainer` (page 2955)
- `setLayoutManager:` (page 2787) (NSTextContainer)
- `replaceLayoutManager:` (page 2785) (NSTextContainer)

Related Sample Code
LayoutManagerDemo
Sketch-112

Declared In
`NSTextView.h`

linkTextAttributes

Returns the attributes used to draw the onscreen presentation of link text.

```
- (NSDictionary *)linkTextAttributes
```

Return Value
A dictionary of attributes corresponding to the onscreen presentation of link text.

Discussion
Link text attributes are applied as temporary attributes to any text with a link attribute. Candidates include those attributes that do not affect layout.

In applications created prior to Mac OS X v10.3, the default value is an empty dictionary. In applications created with Mac OS X v10.3 or greater, the default attributes specify blue text with a single underline and the pointing hand cursor.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setLinkTextAttributes:` (page 2937)

Declared In
`NSTextView.h`

loosenKerning:

Increases the space between glyphs in the receiver's selection, or in all text if the receiver is a plain text view.

```
- (void)loosenKerning:(id)sender
```

Parameters

sender

The control that sent the message; may be `nil`.

Discussion

Kerning values are determined by the point size of the fonts in the selection.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `tightenKerning:` (page 2957)
- `turnOffKerning:` (page 2962)
- `useStandardKerning:` (page 2966)

Declared In

`NSTextView.h`

lowerBaseline:

Lowers the baseline offset of selected text by 1 point, or of all text if the receiver is a plain text view.

```
- (void)lowerBaseline:(id)sender
```

Parameters

sender

The control that sent the message; may be `nil`.

Discussion

As such, this method defines a more primitive operation than subscripting.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `raiseBaseline:` (page 2911)
- `subscript:` (page 2744) (NSText)
- `unscript:` (page 2746) (NSText)

Declared In

`NSTextView.h`

markedTextAttributes

Returns the attributes used to draw marked text.

```
- (NSDictionary *)markedTextAttributes
```

Return Value

A dictionary of attributes used to draw marked text. Text color, background color, and underline are the only supported attributes for marked text.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMarkedTextAttributes:` (page 2938)

Declared In
`NSTextView.h`

orderFrontLinkPanel:

Brings forward a panel allowing the user to manipulate links in the text view.

```
- (void)orderFrontLinkPanel:(id)sender
```

Parameters

sender
The control sending the message. May be `nil`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTextView.h`

orderFrontListPanel:

Brings forward a panel allowing the user to manipulate text lists in the text view.

```
- (void)orderFrontListPanel:(id)sender
```

Parameters

sender
The control sending the message. May be `nil`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTextView.h`

orderFrontSpacingPanel:

Brings forward a panel allowing the user to manipulate text line heights, interline spacing, and paragraph spacing, in the text view.

```
- (void)orderFrontSpacingPanel:(id)sender
```

Parameters

sender
The control sending the message. May be `nil`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTextView.h`

orderFrontSubstitutionsPanel:

Brings forward a panel allowing the user to specify string substitutions in the text view.

```
- (void)orderFrontSubstitutionsPanel:(id)sender
```

Parameters

sender
The control sending the message. May be `nil`.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTextView.h`

orderFrontTablePanel:

Brings forward a panel allowing the user to manipulate text tables in the text view.

```
- (void)orderFrontTablePanel:(id)sender
```

Parameters

sender
The control sending the message. May be `nil`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTextView.h`

outline:

Adds the outline attribute to the selected text attributes if absent; removes the attribute if present.

```
- (void)outline:(id)sender
```

Parameters

sender
The control that sent the message; may be `nil`.

Discussion
If there is a selection and the first character of the selected range has a non-zero stroke width, or if there is no selection and the typing attributes have a non-zero stroke width, then the stroke width is removed; otherwise the value of `NSStrokeWidthAttributeName` is set to the default value for outline (3.0).

Operates on the selected range if the receiver contains rich text. For plain text the range is the entire contents of the receiver.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTextView.h

pasteAsPlainText:

Inserts the contents of the pasteboard into the receiver's text as plain text.

```
- (void)pasteAsPlainText:(id)sender
```

Parameters

sender
The control that sent the message; may be `nil`.

Discussion
This method behaves analogously to `insertText:` (page 2898).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `pasteAsRichText:` (page 2909)
- `insertText:` (page 2898)

Declared In
NSTextView.h

pasteAsRichText:

This action method inserts the contents of the pasteboard into the receiver's text as rich text, maintaining its attributes.

```
- (void)pasteAsRichText:(id)sender
```

Parameters

sender
The control that sent the message; may be `nil`.

Discussion
The text is inserted at the insertion point if there is one, otherwise replacing the selection.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `pasteAsRichText:` (page 2909)
- `insertText:` (page 2898)

Declared In
NSTextView.h

performFindPanelAction:

Performs a find panel action specified by the sender's tag.

```
- (void)performFindPanelAction:(id)sender
```

Parameters

sender

The control sending the message. This method sends the `tag` (page 840) method to determine what operation to perform. The list of possible tags is provided in "Constants" (page 2970).

Discussion
This is the generic action method for the find menu and find panel, and can be overridden to implement a custom find panel.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTextView.h

preferredPasteboardTypeFromArray:restrictedToTypesFromArray:

Returns whatever type on the pasteboard would be most preferred for copying data.

```
- (NSString *)preferredPasteboardTypeFromArray:(NSArray *)availableTypes
    restrictedToTypesFromArray:(NSArray *)allowedTypes
```

Parameters

availableTypes

The types currently available on the pasteboard.

allowedTypes

Types allowed in the return value. If `nil`, any available type is allowed.

Return Value
The preferred type to provide given the available types and the allowed types.

Discussion
You should not need to override this method. You should also not need to invoke it unless you are implementing a new type of pasteboard to handle services other than copy/paste or dragging.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `pasteAsPlainText:` (page 2909)
- `pasteAsRichText:` (page 2909)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSTextView.h

raiseBaseline:

Raises the baseline offset of selected text by 1 point, or of all text if the receiver is a plain text view.

```
- (void)raiseBaseline:(id)sender
```

Parameters

sender
The control that sent the message; may be nil.

Discussion
As such, this method defines a more primitive operation than superscripting.

Availability
Available in Mac OS X v10.0 and later.

See Also
- lowerBaseline: (page 2906)
- superscript: (page 2745) (NSText)
- unscript: (page 2746) (NSText)

Declared In
NSTextView.h

rangeForUserCharacterAttributeChange

Returns the range of characters affected by an action method that changes character (not paragraph) attributes.

```
- (NSRange)rangeForUserCharacterAttributeChange
```

Return Value
The range of characters affected by an action method that changes character (not paragraph) attributes, such as the NSText action method changeFont: (page 2720). For rich text this range is typically the range of the selection. For plain text this range is the entire contents of the receiver. If the receiver isn't editable or doesn't use the Font panel, the range returned has a location of NSNotFound.

Special Considerations

In Mac OS X v10.4 and later, returns the first subrange where there is a multiple-range selection.

Availability
Available in Mac OS X v10.0 and later.

See Also
- rangesForUserCharacterAttributeChange (page 2913)
- rangeForUserParagraphAttributeChange (page 2912)
- rangeForUserTextChange (page 2913)
- isEditable (page 2902)
- usesFontPanel (page 2966)

Declared In
NSTextView.h

rangeForUserCompletion

Returns the partial range from the most recent beginning of a word up to the insertion point.

```
- (NSRange)rangeForUserCompletion
```

Return Value
The partial range from the most recent beginning of a word up to the insertion point. Returning (NSNotFound, 0) suppresses completion.

Discussion
May be overridden by subclasses to alter the range to be completed.

The return value from this method is intended to be used for the range argument in the text completion methods such as completionsForPartialWordRange:indexOfSelectedItem: (page 2888).

Special Considerations

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTextView.h

rangeForUserParagraphAttributeChange

Returns the range of characters affected by an action method that changes paragraph (not character) attributes.

```
- (NSRange)rangeForUserParagraphAttributeChange
```

Return Value
The range of characters affected by an action method that changes paragraph (not character) attributes, such as the NSText action method alignLeft: (page 2718). For rich text this range is typically calculated by extending the range of the selection to paragraph boundaries. For plain text this range is the entire contents of the receiver. If the receiver isn't editable or doesn't use the Font panel, the range returned has a location of NSNotFound.

Special Considerations

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

Availability
Available in Mac OS X v10.0 and later.

See Also
- rangesForUserParagraphAttributeChange (page 2914)
- rangeForUserCharacterAttributeChange (page 2911)

- `rangeForUserTextChange` (page 2913)
- `isEditable` (page 2902)
- `usesRuler` (page 2966)

Declared In
`NSTextView.h`

rangeForUserTextChange

Returns the range of characters affected by a method that changes characters (as opposed to attributes).

```
- (NSRange)rangeForUserTextChange
```

Return Value
The range of characters affected by a method that changes characters (as opposed to attributes), such as `insertText:` (page 2898). This is typically the range of the selection. If the receiver isn't editable the range returned has a location of `NSNotFound`.

Special Considerations

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rangesForUserTextChange` (page 2914)
- `rangeForUserParagraphAttributeChange` (page 2912)
- `rangeForUserCharacterAttributeChange` (page 2911)
- `isEditable` (page 2902)
- `usesRuler` (page 2966)

Declared In
`NSTextView.h`

rangesForUserCharacterAttributeChange

Returns an array containing the ranges of characters affected by an action method that changes character (not paragraph) attributes.

```
- (NSArray *)rangesForUserCharacterAttributeChange
```

Return Value
An array containing the ranges of characters affected by an action method that changes character (not paragraph) attributes, such as the NSText action method `changeFont:` (page 2720). For rich text these ranges are typically the ranges of the selections. For plain text the range is the entire contents of the receiver. Returns `nil` if the receiver isn't editable or doesn't use the Font panel.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `rangesForUserParagraphAttributeChange` (page 2914)
- `rangesForUserTextChange` (page 2914)
- `isEditable` (page 2902)
- `usesFontPanel` (page 2966)

Declared In
`NSTextView.h`

rangesForUserParagraphAttributeChange

Returns an array containing the ranges of characters affected by a method that changes paragraph (not character) attributes.

```
- (NSArray *)rangesForUserParagraphAttributeChange
```

Return Value
An array containing the ranges of characters affected by an action method that changes paragraph (not character) attributes, such as the NSText action method `alignLeft:` (page 2718). For rich text these ranges are typically calculated by extending the range of the selection to paragraph boundaries. For plain text the range is the entire contents of the receiver. Returns `nil` if the receiver isn't editable or doesn't use the Font panel.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `rangesForUserCharacterAttributeChange` (page 2913)
- `rangesForUserTextChange` (page 2914)
- `isEditable` (page 2902)
- `usesRuler` (page 2966)

Declared In
`NSTextView.h`

rangesForUserTextChange

Returns an array containing the ranges of characters affected by a method that changes characters (as opposed to attributes).

```
- (NSArray *)rangesForUserTextChange
```

Return Value
An array containing the ranges of characters affected by a method that changes characters (as opposed to attributes), such as `insertText:` (page 2898). These are typically the ranges of the selections. Returns `nil` if the receiver isn't editable.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `rangesForUserCharacterAttributeChange` (page 2913)

- rangesForUserParagraphAttributeChange (page 2914)
- isEditable (page 2902)
- usesRuler (page 2966)

Declared In
NSTextView.h

readablePasteboardTypes

Returns the types this text view can read immediately from the pasteboard.

```
- (NSArray *)readablePasteboardTypes
```

Return Value
An array of strings describing the types this text view can read immediately from the pasteboard. The strings are ordered by the default preferences.

Discussion
You can override this method to provide support for new types of data. If you want to add support for the default types, you can invoke the superclass version of this method or add the types directly in your overridden version.

Availability
Available in Mac OS X v10.0 and later.

See Also
- preferredPasteboardTypeFromArray:restrictedToTypesFromArray: (page 2910)
- writablePasteboardTypes (page 2968)

Declared In
NSTextView.h

readSelectionFromPasteboard:

Reads the text view's preferred type of data from the specified pasteboard.

```
- (BOOL)readSelectionFromPasteboard:(NSPasteboard *)pboard
```

Parameters
pboard
The pasteboard to read from.

Return Value
YES if the data was successfully read, NO otherwise.

Discussion
This method invokes the preferredPasteboardTypeFromArray:restrictedToTypesFromArray: (page 2910) method to determine the text view's preferred type of data and then reads the data using the readSelectionFromPasteboard:type: (page 2916) method.

You should not need to override this method. You might need to invoke this method if you are implementing a new type of pasteboard to handle services other than copy/paste or dragging.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `preferredPasteboardTypeFromArray:restrictedToTypesFromArray:` (page 2910)
- `readSelectionFromPasteboard:type:` (page 2916)

Declared In
`NSTextView.h`

readSelectionFromPasteboard:type:

Reads data of the given type from the specified pasteboard.

```
- (BOOL)readSelectionFromPasteboard:(NSPasteboard *)pboard type:(NSString *)type
```

Parameters

pboard
The pasteboard to read from.

type
The type of data to read.

Return Value
`YES` if the data was successfully read, `NO` otherwise.

Discussion
The new data is placed at the current insertion point, replacing the current selection if one exists.

You should override this method to read pasteboard types other than the default types. Use the `rangeForUserTextChange` (page 2913) method to obtain the range of characters (if any) to be replaced by the new data.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rangeForUserTextChange` (page 2913)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSTextView.h`

replaceTextContainer:

Replaces the text container for the group of text system objects containing the receiver, keeping the association between the receiver and its layout manager intact.

```
- (void)replaceTextContainer:(NSTextContainer *)aTextContainer
```

Parameters

aTextContainer

The new text container. This method raises `NSInvalidArgumentException` if *aTextContainer* is `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initWithFrame:textContainer:` (page 2896)
- `setTextContainer:` (page 2945)

Related Sample Code
ClipboardViewer

Declared In
`NSTextView.h`

resignFirstResponder

Notifies the receiver that it's been asked to relinquish its status as first responder. (Available in Mac OS X v10.0 through Mac OS X v10.4. Use the `NSWindow` method `makeFirstResponder:` (page 3344) to make a text view the first responder.)

```
- (BOOL)resignFirstResponder
```

Return Value
`YES` if the text view will resign first responder, `NO` otherwise.

Discussion
If the object that will become the new first responder is a text view attached to the same layout manager as the receiver, this method returns `YES` with no further action. Otherwise, this method sends a `textShouldEndEditing:` message to its delegate (if any). If the delegate returns `NO`, this method returns `NO`. If the delegate returns `YES`, this method hides the selection highlighting and posts an `NSTextDidEndEditingNotification` (page 2752) to the default notification center and then returns `YES`.

Use the `NSWindow` method `makeFirstResponder:` (page 3344), not this method, to make a text view the first responder. Never invoke this method directly.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.4.

See Also
- `breakUndoCoalescing` (page 2883)

Declared In
`NSTextView.h`

rulerView:didAddMarker:

Modifies the paragraph style of the paragraphs containing the selection to accommodate a new marker.

```
- (void)rulerView:(NSRulerView *)aRulerView didAddMarker:(NSRulerMarker *)aMarker
```

Parameters

aRulerView

The ruler view sending the message.

aMarker

The marker that was added.

Discussion

This method records the change by invoking `didChangeText` (page 2890) after adding the marker.

`NSTextView` checks for permission to make the change in its `rulerView:shouldAddMarker:` (page 2920) method, which invokes `shouldChangeTextInRange:replacementString:` (page 2949) to send out the proper request and notifications, and only invokes this method if permission is granted.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `representedObject` (page 2239) (NSRulerMarker)
- `rulerView:didMoveMarker:` (page 2918)
- `rulerView:didRemoveMarker:` (page 2919)

Declared In

`NSTextView.h`

rulerView:didMoveMarker:

Modifies the paragraph style of the paragraphs containing the selection to record the new location of the marker.

```
- (void)rulerView:(NSRulerView *)aRulerView didMoveMarker:(NSRulerMarker *)aMarker
```

Parameters

aRulerView

The ruler view sending the message.

aMarker

The marker that was moved.

Discussion

This method records the change by invoking `didChangeText` (page 2890) after moving the marker.

`NSTextView` checks for permission to make the change in its `rulerView:shouldMoveMarker:` (page 2920) method, which invokes `shouldChangeTextInRange:replacementString:` (page 2949) to send out the proper request and notifications, and only invokes this method if permission is granted.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `representedObject` (page 2239) (NSRulerMarker)
- `rulerView:didAddMarker:` (page 2917)
- `rulerView:didRemoveMarker:` (page 2919)

Declared In
NSTextView.h

rulerView:didRemoveMarker:

Modifies the paragraph style of the paragraphs containing the selection—if possible—by removing the specified marker.

```
- (void)rulerView:(NSRulerView *)aRulerView didRemoveMarker:(NSRulerMarker *)aMarker
```

Parameters

aRulerView
The ruler view sending the message.

aMarker
The marker that was removed.

Discussion
This method records the change by invoking `didChangeText` (page 2890) after removing the marker.

`NSTextView` checks for permission to move or remove a tab stop in its `rulerView:shouldMoveMarker:` (page 2920) method, which invokes `shouldChangeTextInRange:replacementString:` (page 2949) to send out the proper request and notifications, and only invokes this method if permission is granted.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `representedObject` (page 2239) (NSRulerMarker)
- `shouldChangeTextInRange:replacementString:` (page 2949)
- `rulerView:didAddMarker:` (page 2917)
- `rulerView:didMoveMarker:` (page 2918)

Declared In
NSTextView.h

rulerView:handleMouseDown:

Adds a left tab marker to the ruler at the location clicked.

```
- (void)rulerView:(NSRulerView *)aRulerView handleMouseDown:(NSEvent *)theEvent
```

Parameters

aRulerView
The ruler view sending the message.

theEvent
The mouse down event.

Discussion
A subclass can override this method to provide other behavior, such as creating guidelines. This method is invoked once with *theEvent* when the user first clicks the ruler area of *aRulerView*, as described in the `NSRulerView` class specification.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextView.h

rulerView:shouldAddMarker:

Returns whether a new marker can be added.

```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldAddMarker:(NSRulerMarker *)aMarker
```

Parameters

aRulerView
The ruler view sending the message.

aMarker
The marker to be added.

Return Value
YES if *aMarker* can be added, NO otherwise.

Discussion
The receiver checks for permission to make the change by invoking `shouldChangeTextInRange:replacementString:` (page 2949) and returning the return value of that message. If the change is allowed, the receiver is then sent a `rulerView:didAddMarker:` (page 2917) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rulerView:shouldMoveMarker:` (page 2920)
- `rulerView:shouldRemoveMarker:` (page 2921)

Declared In
NSTextView.h

rulerView:shouldMoveMarker:

Returns whether the marker should be moved.

```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldMoveMarker:(NSRulerMarker *)aMarker
```

Parameters

aRulerView
The ruler view sending the message.

aMarker
The marker to be moved.

Return Value
YES if *aMarker* can be moved, NO otherwise.

Discussion
This method controls whether an existing marker *aMarker* can be moved. The receiver checks for permission to make the change by invoking `shouldChangeTextInRange:replacementString:` (page 2949) and returning the return value of that message. If the change is allowed, the receiver is then sent a `rulerView:didMoveMarker:` (page 2918) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rulerView:shouldAddMarker:` (page 2920)
- `rulerView:shouldRemoveMarker:` (page 2921)

Declared In
`NSTextView.h`

rulerView:shouldRemoveMarker:

Returns whether the marker should be removed.

```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldRemoveMarker:(NSRulerMarker
    *)aMarker
```

Parameters

aRulerView
The ruler view sending the message.

aMarker
The marker to be removed.

Return Value
`YES` if *aMarker* can be removed, `NO` otherwise.

Discussion
Only markers that represent tab stops can be removed. This method returns `YES` if *aMarker* represents an `NSTextTab` object, `NO` otherwise. Because this method can be invoked repeatedly as the user drags a ruler marker, it returns that value immediately. If the change is allowed and the user actually removes the marker, the receiver is also sent a `rulerView:didRemoveMarker:` (page 2919) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rulerView:shouldAddMarker:` (page 2920)
- `rulerView:shouldMoveMarker:` (page 2920)

Declared In
`NSTextView.h`

rulerView:willAddMarker:atLocation:

Returns a potentially modified location to which the marker should be added.

```
- (CGFloat)rulerView:(NSRulerView *)aRulerView willAddMarker:(NSRulerMarker *)aMarker
    atLocation:(CGFloat)location
```

Parameters

aRulerView

The ruler view sending the message.

aMarker

The marker to be added.

location

The new location for the marker.

Return Value

The modified location to which the marker should be added.

Discussion

This method ensures that the proposed *location* of *aMarker* lies within the appropriate bounds for the receiver's text container, returning the modified location.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `rulerView:didAddMarker:` (page 2917)

Declared In

`NSTextView.h`

rulerView:willMoveMarker:toLocation:

Returns a potentially modified location to which the marker should be moved.

```
- (CGFloat)rulerView:(NSRulerView *)aRulerView willMoveMarker:(NSRulerMarker
    *)aMarker toLocation:(CGFloat)location
```

Parameters

aRulerView

The ruler view sending the message.

aMarker

The marker to be moved.

location

The new location for the marker.

Return Value

The modified location to which the marker should be moved.

Discussion

This method ensures that the proposed *location* of *aMarker* lies within the appropriate bounds for the receiver's text container.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `rulerView:didMoveMarker:` (page 2918)

Declared In
`NSTextView.h`

selectedRanges

Returns an array containing the ranges of characters selected in the receiver's layout manager.

```
- (NSArray *)selectedRanges
```

Return Value
A non-nil, non-empty array of objects responding to the NSValue `rangeValue` method. In addition, the objects in the array are sorted, non-overlapping, non-contiguous, and (except for the case of a single range) have non-zero-length.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setSelectedRanges:` (page 2942)

Declared In
`NSTextView.h`

selectedTextAttributes

Returns the attributes used to indicate the selection.

```
- (NSDictionary *)selectedTextAttributes
```

Return Value
A dictionary of attributes used to indicate the selection. Text color, background color, and underline are the only supported attributes for selected text. Typically only the text background color is used.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedRange` (page 3866) (NSTextInput)
- `setSelectedTextAttributes:` (page 2943)

Declared In
`NSTextView.h`

selectionAffinity

Returns the preferred direction of selection.

```
- (NSSelectionAffinity)selectionAffinity
```

Return Value
The preferred direction of selection.

Discussion
Selection affinity determines whether, for example, the insertion point appears after the last character on a line or before the first character on the following line in cases where text wraps across line boundaries.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setSelectedRange:affinity:stillSelecting:` (page 2941)

Declared In
`NSTextView.h`

selectionGranularity

Returns the current selection granularity, used during mouse tracking to modify the range of the selection.

```
- (NSSelectionGranularity)selectionGranularity
```

Return Value
The current selection granularity.

Discussion
See `setSelectionGranularity:` (page 2944) for a discussion of how selection granularity affects the behavior of selection extension.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectionRangeForProposedRange:granularity:` (page 2924)
- `setSelectionGranularity:` (page 2944)

Declared In
`NSTextView.h`

selectionRangeForProposedRange:granularity:

Returns an adjusted selected range based on the selection granularity.

```
- (NSRange)selectionRangeForProposedRange:(NSRange)proposedSelRange
    granularity:(NSSelectionGranularity)granularity
```

Parameters

proposedSelRange
The proposed selected range.

granularity
The selection granularity.

Return Value
The adjusted selected range, taking into account the selection granularity.

Discussion
This method is invoked repeatedly during mouse tracking to modify the range of the selection. Override this method to specialize selection behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setSelectionGranularity:` (page 2944)

Declared In
`NSTextView.h`

setAcceptsGlyphInfo:

Sets whether the receiver accepts the glyph info attribute.

```
- (void)setAcceptsGlyphInfo:(BOOL)flag
```

Parameters

flag

`YES` if the receiver should accept the `NSGlyphInfoAttributeName` attribute from text input sources such as input methods and the pasteboard, `NO` otherwise.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- acceptsGlyphInfo` (page 2880)

Declared In
`NSTextView.h`

setAlignment:range:

Sets the alignment of the paragraphs containing characters in the specified range.

```
- (void)setAlignment:(NSTextAlignment)alignment range:(NSRange)aRange
```

Parameters

alignment

The new alignment.

aRange

The range of characters whose paragraphs will have their alignment set.

Discussion
This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 2950) or `shouldChangeTextInRange:replacementString:` (page 2949) to include this method in an undoable action.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rangeForUserParagraphAttributeChange` (page 2912)

Declared In
`NSTextView.h`

setAllowedInputSourceLocales:

Sets an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.

```
- (void)setAllowedInputSourceLocales:(NSArray *)localeIdentifiers
```

Parameters

localeIdentifiers
The new locale identifiers of allowed input sources.

Discussion
You can use the meta-locale identifier, `NSAllRomanInputSourcesLocaleIdentifier` (page 2972), to specify input sources that are limited for Roman script editing.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `allowedInputSourceLocales` (page 2881)

Declared In
`NSTextView.h`

setAllowsDocumentBackgroundColorChange:

Sets whether the receiver allows its background color to change.

```
- (void)setAllowsDocumentBackgroundColorChange:(BOOL)flag
```

Parameters

flag
`YES` if the receiver allows the background color to change, otherwise `NO`.

Discussion
This corresponds to the background color of the entirety of the text view, not just to a selected range of text.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `allowsDocumentBackgroundColorChange` (page 2881)
- `changeDocumentBackgroundColor:` (page 2884)

Related Sample Code
CIAnnotation
Quartz Composer WWDC 2005 TextEdit

Declared In
NSTextView.h

setAllowsImageEditing:

Specifies whether image attachments should permit editing of their images.

```
- (void)setAllowsImageEditing:(BOOL)flag
```

Parameters

flag

If YES, image editing is allowed; if NO, it is not allowed.

Discussion
For image editing to be allowed, the text view must be editable and the text attachment cell must support image editing.

Availability
Available in Mac OS X v10.5 and later.

See Also
- allowsImageEditing (page 2881)

Declared In
NSTextView.h

setAllowsUndo:

Sets whether undo support is enabled.

```
- (void)setAllowsUndo:(BOOL)flag
```

Parameters

flag

YES to enable undo support, NO otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- allowsUndo (page 2882)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
QuickLookSketch
Sketch+Accessibility
Sketch-112
TextSizingExample

Declared In
NSTextView.h

setAutomaticDashSubstitutionEnabled:

Sets whether automatic dash substitution is enabled.

```
- (void)setAutomaticDashSubstitutionEnabled:(BOOL)flag
```

Parameters

flag

YES if it should be enabled, otherwise NO.

Discussion
Turning on automatic dash substitution enables automatic conversion of sequences of ASCII hyphen (-) characters to typographic dashes.

Availability
Available in Mac OS X v10.6 and later.

See Also
- isAutomaticDashSubstitutionEnabled (page 2899)
- toggleAutomaticDashSubstitution: (page 2957)

Declared In
NSTextView.h

setAutomaticDataDetectionEnabled:

Sets whether automatic data detection is enabled.

```
- (void)setAutomaticDataDetectionEnabled:(BOOL)flag
```

Parameters

flag

YES if it should be enabled, otherwise NO.

Discussion
Automatic data detection enables detection of dates, addresses, and phone numbers.

Availability
Available in Mac OS X v10.6 and later.

See Also
- isAutomaticDataDetectionEnabled (page 2899)
- toggleAutomaticDataDetection: (page 2958)

Declared In
NSTextView.h

setAutomaticLinkDetectionEnabled:

Enables or disables automatic link detection.

```
- (void)setAutomaticLinkDetectionEnabled:(BOOL)flag
```

Parameters

flag

If YES, automatic link detection is enabled; if NO, it is disabled.

Discussion

Automatic link detection causes strings representing URLs typed in the view to be automatically made into links to those URLs.

Availability

Available in Mac OS X v10.5 and later.

See Also

- isAutomaticLinkDetectionEnabled (page 2899)
- toggleAutomaticLinkDetection: (page 2958)

URLAtIndex:effectiveRange: (page 270) (NSAttributedString)

Declared In

NSTextView.h

setAutomaticQuoteSubstitutionEnabled:

Enables and disables automatic quotation mark substitution.

```
- (void)setAutomaticQuoteSubstitutionEnabled:(BOOL)flag
```

Parameters

flag

If YES, automatic quotation mark substitution is enabled; if NO, it is disabled.

Discussion

Automatic quote substitution causes ASCII quotation marks and apostrophes to be automatically replaced, on a context-dependent basis, with more typographically accurate symbols.

Availability

Available in Mac OS X v10.5 and later.

See Also

- isAutomaticQuoteSubstitutionEnabled (page 2900)
- toggleAutomaticQuoteSubstitution: (page 2959)

Declared In

NSTextView.h

setAutomaticSpellingCorrectionEnabled:

Sets whether automatic spelling correction is enabled.

```
- (void)setAutomaticSpellingCorrectionEnabled:(BOOL)flag
```

Parameters

flag

YES if it should be enabled, otherwise NO.

Availability
Available in Mac OS X v10.6 and later.

See Also
- isAutomaticSpellingCorrectionEnabled (page 2900)
- toggleAutomaticSpellingCorrection: (page 2959)

Declared In
NSTextView.h

setAutomaticTextReplacementEnabled:

Sets whether automatic text replacement is enabled.

```
- (void)setAutomaticTextReplacementEnabled:(BOOL)flag
```

Parameters

flag

YES if it should be enabled, otherwise NO.

Discussion
Turning on automatic text replacement enables automatic substitution of a variety of static text items based on user preferences.

Availability
Available in Mac OS X v10.6 and later.

See Also
- isAutomaticTextReplacementEnabled (page 2901)
- toggleAutomaticTextReplacement: (page 2960)

Declared In
NSTextView.h

setBackgroundColor:

Sets the receiver's background color.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Parameters

aColor

The new background color.

Special Considerations

This method does not include undo support by default. Clients must invoke shouldChangeTextInRanges:replacementStrings: (page 2950) or shouldChangeTextInRange:replacementString: (page 2949) to include this method in an undoable action.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDrawsBackground:` (page 2934)
- `backgroundColor` (page 2882)

Related Sample Code
CIAnnotation
OpenCL NBody Simulation Example
Quartz Composer WWDC 2005 TextEdit
TextSizingExample

Declared In
`NSTextView.h`

setBaseWritingDirection:range:

Sets the base writing direction of a range of text.

```
- (void)setBaseWritingDirection:(NSWritingDirection)writingDirection
    range:(NSRange)range
```

Parameters

writingDirection
The new writing direction for the text in *range*.

range
The range of text that will have the new writing direction.

Discussion
Invoke this method to change the base writing direction from left-to-right to right-to-left for languages like Hebrew and Arabic, for example.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 2950) or `shouldChangeTextInRange:replacementString:` (page 2949) to include this method in an undoable action.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTextView.h`

setConstrainedFrameSize:

Attempts to set the frame size as if by user action.

```
- (void)setConstrainedFrameSize:(NSSize)desiredSize
```

Parameters

desiredSize

The new desired size.

Discussion

This method respects the receiver's existing minimum and maximum sizes and by whether resizing is permitted.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `minSize` (page 2727) (NSText)
- `maxSize` (page 2727) (NSText)
- `isHorizontallyResizable` (page 2725) (NSText)
- `isVerticallyResizable` (page 2727) (NSText)

Related Sample Code

CIAnnotation

Declared In

`NSTextView.h`

setContinuousSpellCheckingEnabled:

Enables or disables continuous spell checking.

```
- (void)setContinuousSpellCheckingEnabled:(BOOL)flag
```

Parameters

flag

If `YES`, enables continuous spell checking; if `NO`, disables it.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `isContinuousSpellCheckingEnabled` (page 2902)
- `toggleContinuousSpellChecking:` (page 2960)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

`NSTextView.h`

setDefaultParagraphStyle:

Sets the receiver's default paragraph style.

```
- (void)setDefaultParagraphStyle:(NSParagraphStyle *)paragraphStyle
```

Parameters

paragraphStyle

The new default paragraph style.

Availability

Available in Mac OS X v10.3 and later.

See Also

- defaultParagraphStyle (page 2889)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

NSTextView.h

setDelegate:

Sets the delegate for all text views sharing the receiver's layout manager.

```
- (void)setDelegate:(id < NSTextViewDelegate >)anObject
```

Parameters

anObject

The new delegate object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- delegate (page 2889)

Related Sample Code

CIAnnotation

FunHouse

Quartz Composer WWDC 2005 TextEdit

Sketch-112

TextSizingExample

Declared In

NSTextView.h

setDisplaysLinkToolTips:

Enables or disables automatic display of link tooltips.

```
- (void)setDisplaysLinkToolTips:(BOOL)flag
```

Parameters

flag

If YES, automatic link tooltip display is enabled; if NO, it is disabled.

Discussion
The default value for this feature is `YES`; clients who do not wish tooltips to be displayed automatically must explicitly disable it.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- displaysLinkToolTips` (page 2890)

Declared In
`NSTextView.h`

setDrawsBackground:

Sets whether the receiver draws its background.

```
- (void)setDrawsBackground:(BOOL)flag
```

Parameters

flag

`YES` to cause the receiver to fill its background with the background color, `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setBackgroundColor:` (page 2930)
`- drawsBackground` (page 2893)

Related Sample Code
CIAnnotation
QuickLookSketch
Sketch+Accessibility
Sketch-112
TextSizingExample

Declared In
`NSTextView.h`

setEditable:

Controls whether the text views sharing the receiver's layout manager allow the user to edit text.

```
- (void)setEditable:(BOOL)flag
```

Parameters

flag

`YES` to allow the user to edit text and attributes of all text views sharing the receiver's layout manager, `NO` otherwise.

Discussion
If a text view is made editable, it's also made selectable. Text views are editable by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setSelectable:` (page 2940)
- `isEditable` (page 2902)

Related Sample Code
FunHouse
GLUT
LSMSmartCategorizer
Quartz Composer WWDC 2005 TextEdit
TextSizingExample

Declared In
`NSTextView.h`

setEnabledTextCheckingTypes:

Sets the default text checking types.

```
- (void)setEnabledTextCheckingTypes:(NSTextCheckingTypes)checkingTypes
```

Parameters
checkingTypes
The types of text checking to perform by default. See `NSTextCheckingTypes` for possible values.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `enabledTextCheckingTypes` (page 2894)

Declared In
`NSTextView.h`

setFieldEditor:

Controls whether the text views sharing the receiver's layout manager behave as field editors.

```
- (void)setFieldEditor:(BOOL)flag
```

Parameters
flag
`YES` to cause the text views sharing the receiver's layout manager to behave as field editors, `NO` otherwise.

Discussion
Field editors interpret Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder. Non-field editors instead accept these characters as text input. See Text Fields, Text Views, and the Field Editor for more information on field editors. By default, text views don't behave as field editors.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isFieldEditor` (page 2902)

Related Sample Code
CIAnnotation
TextSizingExample

Declared In
`NSTextView.h`

setGrammarCheckingEnabled:

Enables and disables grammar checking.

```
- (void)setGrammarCheckingEnabled:(BOOL)flag
```

Parameters

flag

If `YES`, grammar checking is enabled; if `NO`, it is disabled.

Discussion
If grammar checking is enabled, then it is performed alongside spell checking, whenever the text view checks spelling, whether continuously or manually.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `isGrammarCheckingEnabled` (page 2903)
- `toggleGrammarChecking:` (page 2961)
- `setSpellingState:range:` (page 2945)

Declared In
`NSTextView.h`

setImportsGraphics:

Controls whether the text views sharing the receiver's layout manager allow the user to import files by dragging.

```
- (void)setImportsGraphics:(BOOL)flag
```

Parameters

flag

`YES` to allow the user to import files by dragging onto the text views sharing the receiver's layout manager, `NO` otherwise.

Discussion
Text views that are set to accept dragged files are also set to allow rich text. By default, text views don't accept dragged files but do allow rich text.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textStorage` (page 2957)
- `setRichText:` (page 2939)
- `importsGraphics` (page 2895)

Related Sample Code
CIAnnotation
FunHouse
GLUT
Quartz Composer WWDC 2005 TextEdit
TextSizingExample

Declared In
`NSTextView.h`

setInsertionPointColor:

Sets the color of the insertion point

```
- (void)setInsertionPointColor:(NSColor *)aColor
```

Parameters
aColor
The new color.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawInsertionPointInRect:color:turnedOn:` (page 2892)
- `shouldDrawInsertionPoint` (page 2950)
- `insertionPointColor` (page 2897)

Declared In
`NSTextView.h`

setLinkTextAttributes:

Sets the attributes used to draw the onscreen presentation of link text.

```
- (void)setLinkTextAttributes:(NSDictionary *)attributeDictionary
```

Parameters
attributeDictionary
A dictionary of attributes corresponding to the onscreen presentation of link text.

Discussion
Link text attributes are applied as temporary attributes to any text with a link attribute. Candidates include those attributes that do not affect layout.

In applications created prior to Mac OS X v10.3, the default value is an empty dictionary. In applications created with Mac OS X v10.3 or greater, the default attributes specify blue text with a single underline and the pointing hand cursor.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `linkTextAttributes` (page 2905)

Declared In
`NSTextView.h`

setMarkedTextAttributes:

Sets the attributes used to draw marked text.

```
- (void)setMarkedTextAttributes:(NSDictionary *)attributes
```

Parameters

attributes

A dictionary of attributes used to draw marked text. Text color, background color, and underline are the only supported attributes for marked text.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `markedTextAttributes` (page 2906)
- `markedRange` (page 3865) (NSTextInput)

Declared In
`NSTextView.h`

setNeedsDisplayInRect:avoidAdditionalLayout:

Marks the receiver as requiring display.

```
- (void)setNeedsDisplayInRect:(NSRect)aRect avoidAdditionalLayout:(BOOL)flag
```

Parameters

aRect

The rectangle in which display is required.

flag

A value of `YES` causes the receiver to not perform any layout, even if this means that portions of the text view remain empty. Otherwise the receiver performs at least as much layout as needed to display *aRect*.

Discussion
`NSTextView` overrides the `NSView` `setNeedsDisplayInRect:` (page 3225) method to invoke this method with a *flag* argument of `NO`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextView.h

setRichText:

Controls whether the text views sharing the receiver's layout manager allow the user to apply attributes to specific ranges of text.

```
- (void)setRichText:(BOOL)flag
```

Parameters

flag

YES to allow the user to apply attributes to specific ranges of text in text views sharing the receiver's layout manager, NO otherwise.

Discussion
Text fields that don't allow rich text also don't accept dragged files. By default, text views let the user apply multiple attributes to text, but don't accept dragged files.

Availability
Available in Mac OS X v10.0 and later.

See Also
- textStorage (page 2957)
- isRichText (page 2903)
- setImportsGraphics: (page 2936)

Related Sample Code
CIAnnotation
FunHouse
Quartz Composer WWDC 2005 TextEdit
TextSizingExample
TipWrapper

Declared In
NSTextView.h

setRulerVisible:

Controls whether the scroll view enclosing text views sharing the receiver's layout manager displays the ruler.

```
- (void)setRulerVisible:(BOOL)flag
```

Parameters

flag

YES to show the ruler, NO to hide the ruler. By default, the ruler is hidden.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setUsesRuler:` (page 2948)
- `isRulerVisible` (page 2904)
- `toggleRuler:` (page 2745) (NSText)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSTextView.h`

setSelectable:

Controls whether the text views sharing the receiver's layout manager allow the user to select text.

```
- (void)setSelectable:(BOOL)flag
```

Parameters

flag

`YES` to allow the user to select text of all text views sharing the receiver's layout manager; otherwise, `NO`.

Discussion
If a text view is made not selectable, it's also made not editable, and buttons on the Find panel are dimmed. Text views are by default both editable and selectable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setEditable:` (page 2934)
- `isSelectable` (page 2904)

Related Sample Code
CIAnnotation
FunHouse
GLUT
TextSizingExample

Declared In
`NSTextView.h`

setSelectedRange:

Sets the selection to the characters in a single range.

```
- (void)setSelectedRange:(NSRange)charRange
```

Parameters

charRange

The range of characters to select. This range must begin and end on glyph boundaries and not split base glyphs and their nonspacing marks.

Discussion

This method sets the selection to the characters in *charRange*, resets the selection granularity to `NSSelectByCharacter`, and posts an `NSTextViewDidChangeSelectionNotification` (page 2974) to the default notification center. It also removes the marking from marked text if the new selection is greater than the marked region.

Special Considerations

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setSelectedRange:affinity:stillSelecting:` (page 2941)
- `selectionAffinity` (page 2923)
- `selectionGranularity` (page 2924)
- `selectedRange` (page 3866) (NSTextInput)

Related Sample Code

CocoaSpeechSynthesisExample
Quartz Composer WWDC 2005 TextEdit
SBSetFinderComment
Sketch-112
TextSizingExample

Declared In

`NSTextView.h`

setSelectedRange:affinity:stillSelecting:

Sets the selection to a range of characters in response to user action.

```
- (void)setSelectedRange:(NSRange)charRange affinity:(NSSelectionAffinity)affinity
    stillSelecting:(BOOL)flag
```

Parameters

charRange

The range of characters to select. This range must begin and end on glyph boundaries and not split base glyphs and their nonspacing marks.

affinity

The selection affinity for the selection. See `selectionAffinity` (page 2923) for more information about how affinities work.

flag

YES to behave appropriately for a continuing selection where the user is still dragging the mouse, NO otherwise. If YES, the receiver doesn't send notifications or remove the marking from its marked text. If NO, the receiver posts an NSTextViewDidChangeSelectionNotification (page 2974) to the default notification center and removes the marking from marked text if the new selection is greater than the marked region.

Discussion
This method resets the selection granularity to NSSelectByCharacter.

Special Considerations

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setSelectedRange: (page 2940)
- selectionAffinity (page 2923)
- selectionGranularity (page 2924)
- selectedRange (page 3866) (NSTextInput)

Declared In
NSTextView.h

setSelectedRanges:

Sets the selection to the characters in an array of ranges.

```
- (void)setSelectedRanges:(NSArray *)ranges
```

Parameters

ranges

A non-nil, non-empty array of objects responding to the NSValue rangeValue method. The ranges in the *ranges* array must begin and end on glyph boundaries and not split base glyphs and their nonspacing marks.

Discussion
Sets the selection to the characters in the *ranges* array, resets the selection granularity to NSSelectByCharacter, and posts an NSTextViewDidChangeSelectionNotification (page 2974) to the default notification center. Also removes the marking from marked text if the new selection is greater than the marked region.

Availability
Available in Mac OS X v10.4 and later.

See Also
- selectedRanges (page 2923)
- setSelectedRanges:affinity:stillSelecting: (page 2943)
- selectionAffinity (page 2923)
- selectionGranularity (page 2924)

Declared In
NSTextView.h

setSelectedRanges:affinity:stillSelecting:

Sets the selection to the characters in an array of ranges in response to user action.

```
- (void)setSelectedRanges:(NSArray *)ranges affinity:(NSSelectionAffinity)affinity
    stillSelecting:(BOOL)stillSelectingFlag
```

Parameters

ranges

A non-nil, non-empty array of objects responding to the NSValue `rangeValue` method. The ranges in the *ranges* array must begin and end on glyph boundaries and not split base glyphs and their nonspacing marks.

affinity

The selection affinity for the selection. See `selectionAffinity` (page 2923) for more information about how affinities work.

stillSelectingFlag

`YES` to behave appropriately for a continuing selection where the user is still dragging the mouse, `NO` otherwise. If `YES`, the receiver doesn't send notifications or remove the marking from its marked text. If `NO`, the receiver posts an `NSTextViewDidChangeSelectionNotification` (page 2974) to the default notification center and removes the marking from marked text if the new selection is greater than the marked region.

Discussion
This method also resets the selection granularity to `NSSelectByCharacter`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `selectedRanges` (page 2923)
- `setSelectedRanges:` (page 2942)
- `selectionAffinity` (page 2923)
- `selectionGranularity` (page 2924)

Declared In
NSTextView.h

setSelectedTextAttributes:

Sets the attributes used to indicate the selection.

```
- (void)setSelectedTextAttributes:(NSDictionary *)attributes
```

Parameters

attributes

A dictionary of attributes used to indicate the selection. Text color, background color, and underline are the only supported attributes for selected text.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedRange` (page 3866) (NSTextInput)
- `selectedTextAttributes` (page 2923)

Related Sample Code
TextSizingExample

Declared In
`NSTextView.h`

setSelectionGranularity:

Sets the selection granularity for subsequent extension of a selection.

```
- (void)setSelectionGranularity:(NSSelectionGranularity)granularity
```

Parameters

granularity
The new granularity for selection extension.

Discussion
Selection granularity is used to determine how the selection is modified when the user Shift-clicks or drags the mouse after a double or triple click. For example, if the user selects a word by double-clicking, the selection granularity is set to `NSSelectByWord`. Subsequent Shift-clicks then extend the selection by words.

Selection granularity is reset to `NSSelectByCharacter` whenever the selection is set. You should always set the selection granularity after setting the selection.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectionGranularity` (page 2924)
- `setSelectedRange:` (page 2940)

Declared In
`NSTextView.h`

setSmartInsertDeleteEnabled:

Controls whether the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation.

```
- (void)setSmartInsertDeleteEnabled:(BOOL)flag
```

Parameters

flag
`YES` if the receiver should insert or delete space around selected words so as to preserve proper spacing and punctuation, `NO` if it should insert and delete exactly what's selected.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `smartInsertForString:replacingRange:beforeString:afterString:` (page 2953)
- `smartDeleteRangeForProposedRange:` (page 2951)
- `smartInsertDeleteEnabled` (page 2953)

Declared In
`NSTextView.h`

setSpellingState:range:

Sets the spelling state, which controls the display of the spelling and grammar indicators on the given text range.

```
- (void)setSpellingState:(NSInteger)value range:(NSRange)charRange
```

Parameters

value
The spelling state value to set. Possible values, for the temporary attribute on the layout manager using the key `NSSpellingStateAttributeName`, are:

`NSSpellingStateSpellingFlag` (page 287) to highlight spelling issues.

`NSSpellingStateGrammarFlag` (page 287) to highlight grammar issues.

charRange
The character range over which to set the given spelling state.

Discussion
May be called or overridden to control setting of spelling and grammar indicators on text, used to highlight portions of the text that are flagged for spelling or grammar issues.

Calls the method `setEnabledTextCheckingTypes:` (page 2935).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTextView.h`

setTextContainer:

Sets the receiver's text container.

```
- (void)setTextContainer:(NSTextContainer *)aTextContainer
```

Parameters

aTextContainer
The new text container.

Discussion
The receiver uses the layout manager and text storage of *aTextContainer*.

Special Considerations

This method is invoked automatically when you create a text view; you should never invoke it directly, but might want to override it. To change the text view for an established group of text system objects, send `setTextView:` (page 2788) to the text container. To replace the text container for a text view and maintain the view's association with the existing layout manager and text storage, use `replaceTextContainer:` (page 2916).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `textContainer` (page 2955)

Declared In

`NSTextView.h`

setTextContainerInset:

Sets the empty space the receiver leaves around its associated text container.

```
- (void)setTextContainerInset:(NSSize)inset
```

Parameters

inset

The empty space to leave around the text view's text container.

Discussion

It is possible to set the text container and view sizes and resizing behavior so that the inset cannot be maintained exactly, although the text system tries to maintain the inset wherever possible. In any case, the `textContainerOrigin` (page 2956) and size of the text container are authoritative as to the location of the text container within the view.

The text itself can have an additional inset, inside the text container, specified by the `setLineFragmentPadding:` (page 2788) method of `NSTextContainer`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `textContainerOrigin` (page 2956)
- `invalidateTextContainerOrigin` (page 2898)
- `textContainerInset` (page 2956)

Related Sample Code

Sketch-112

TextSizingExample

Declared In

`NSTextView.h`

setTypingAttributes:

Sets the receiver's typing attributes.

```
- (void)setTypingAttributes:(NSDictionary *)attributes
```

Parameters

attributes

A dictionary of the new typing attributes.

Discussion

Typing attributes are reset automatically whenever the selection changes. However, if you add any user actions that change text attributes, the action should use this method to apply those attributes afterwards. User actions that change attributes should always set the typing attributes because there might not be a subsequent change in selection before the next typing.

Availability

Available in Mac OS X v10.0 and later.

See Also

- typingAttributes (page 2963)

Related Sample Code

OpenCL NBody Simulation Example

Quartz Composer WWDC 2005 TextEdit

Declared In

NSTextView.h

setUsesFindPanel:

Specifies whether the receiver allows for a find panel.

```
- (void)setUsesFindPanel:(BOOL)flag
```

Parameters

flag

YES to allow the use of a find panel, NO otherwise.

Availability

Available in Mac OS X v10.3 and later.

See Also

- usesFindPanel (page 2965)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

NSTextView.h

setUsesFontPanel:

Controls whether the text views sharing the receiver's layout manager use the Font panel and Font menu.

```
- (void)setUsesFontPanel:(BOOL)flag
```

Parameters

flag

YES to make the text views sharing the receiver's layout manager respond to messages from the Font panel and from the Font menu, and update the Font panel with the selection font whenever it changes, NO to disallow character attribute changes.

Discussion
By default, text view objects use the Font panel and menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
- rangeForUserCharacterAttributeChange (page 2911)
- usesFontPanel (page 2966)

Related Sample Code
FunHouse
GLUT
Quartz Composer WWDC 2005 TextEdit
TextSizingExample

Declared In
NSTextView.h

setUsesRuler:

Controls whether the text views sharing the receiver's layout manager use a ruler.

```
- (void)setUsesRuler:(BOOL)flag
```

Parameters

flag

YES to cause text views sharing the receiver's layout manager to respond to NSRulerView client messages and to paragraph-related menu actions, and update the ruler (when visible) as the selection changes with its paragraph and tab attributes, otherwise NO.

Discussion
Text views must use a ruler to respond to Format menu commands. If a set of text views don't use the ruler, the ruler is hidden, and the text views disallow paragraph attribute changes. By default, text view objects use the ruler.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setRulerVisible: (page 2939)
- rangeForUserParagraphAttributeChange (page 2912)
- usesRuler (page 2966)

Related Sample Code
FunHouse

Quartz Composer WWDC 2005 TextEdit
TextSizingExample

Declared In
NSTextView.h

shouldChangeTextInRange:replacementString:

Initiates a series of delegate messages (and general notifications) to determine whether modifications can be made to the characters and attributes of the receiver's text.

```
- (BOOL)shouldChangeTextInRange:(NSRange)affectedCharRange
    replacementString:(NSString *)replacementString
```

Parameters

affectedCharRange

The range of characters affected by the proposed change.

replacementString

The characters that will replace those in *affectedCharRange*. If only text attributes are being changed, *replacementString* is nil.

Return Value
YES to allow the change, NO to prohibit it.

Discussion
This method checks with the delegate as needed using textShouldBeginEditing: (page 3857) and textView:shouldChangeTextInRange:replacementString: (page 3889).

This method must be invoked at the start of any sequence of user-initiated editing changes. If your subclass of NSTextView implements new methods that modify the text, make sure to invoke this method to determine whether the change should be made. If the change is allowed, complete the change by invoking the didChangeText (page 2890) method. If you can't determine the affected range or replacement string before beginning changes, pass (NSNotFound, 0) and nil for these values.

Special Considerations

If the receiver is not editable, this method automatically returns NO. This result prevents instances in which a text view could be changed by user actions even though it had been set to be non-editable.

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isEditable (page 2902)
- shouldChangeTextInRanges:replacementStrings: (page 2950)

Declared In
NSTextView.h

shouldChangeTextInRanges:replacementStrings:

Initiates a series of delegate messages (and general notifications) to determine whether modifications can be made to the characters and attributes of the receiver's text.

```
- (BOOL)shouldChangeTextInRanges:(NSArray *)affectedRanges
    replacementStrings:(NSArray *)replacementStrings
```

Parameters

affectedRanges

An array of ranges to change.

replacementStrings

An array of strings containing the characters that replace those in *affectedRanges*, one for each range. If only text attributes are being changed, *replacementStrings* is `nil`.

Return Value

`YES` to allow the change, `NO` to prohibit it.

Discussion

This method checks with the delegate as needed using `textShouldBeginEditing:` and `textView:shouldChangeTextInRanges:replacementStrings:` (page 3889).

This method must be invoked at the start of any sequence of user-initiated editing changes. If your subclass of `NSTextView` implements new methods that modify the text, make sure to invoke this method to determine whether the change should be made. If the change is allowed, complete the change by invoking the `didChangeText` (page 2890) method. If you can't determine the affected range or replacement string before beginning changes, pass `nil` for these values.

Special Considerations

If the receiver is not editable, this method automatically returns `NO`. This result prevents instances in which a text view could be changed by user actions even though it had been set to be non-editable.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- isEditable` (page 2902)

Declared In

`NSTextView.h`

shouldDrawInsertionPoint

Returns whether the receiver should draw its insertion point.

```
- (BOOL)shouldDrawInsertionPoint
```

Return Value

`YES` if the receiver should draw its insertion point, `NO` otherwise.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `drawInsertionPointInRect:color:turnedOn:` (page 2892)

Declared In
`NSTextView.h`

showFindIndicatorForRange:

Causes a temporary highlighting effect to appear around the visible portion (or portions) of the specified range.

```
- (void)showFindIndicatorForRange:(NSRange)charRange
```

Parameters

charRange

The character range around which indicators appear.

Discussion
This method supports lozenge-style indication of find results. The indicators automatically disappear after a certain period of time, or when the method is called again, or when any of a number of changes occur to the view (such as changes to text, view size, or view position).

This method does not itself scroll the specified range to be visible; any desired scrolling should be done before this method is called, first, because the method acts only on the visible portion of the specified range, and, second, because scrolling causes the indicators to disappear. Calling this method with a zero-length range always removes any existing indicators.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTextView.h`

smartDeleteRangeForProposedRange:

Returns an extended range that includes adjacent whitespace that should be deleted along with the proposed range in order to preserve proper spacing and punctuation.

```
- (NSRange)smartDeleteRangeForProposedRange:(NSRange)proposedCharRange
```

Parameters

proposedCharRange

The proposed character range for deleting.

Return Value
An extended range that includes adjacent whitespace that should be deleted along with the proposed range in order to preserve proper spacing and punctuation of the text surrounding the deletion.

Discussion
`NSTextView` uses this method as necessary; you can also use it in implementing your own methods that delete text, typically when the selection granularity is `NSSelectByWord`. To do so, invoke this method with the proposed range to delete, then actually delete the range returned. If placing text on the pasteboard, however, you should put only the characters from the proposed range onto the pasteboard.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `smartInsertForString:replacingRange:beforeString:afterString:` (page 2953)
- `selectionGranularity` (page 2924)
- `smartInsertDeleteEnabled` (page 2953)

Declared In
`NSTextView.h`

smartInsertAfterStringForString:replacingRange:

Returns any whitespace that needs to be added after the string to preserve proper spacing and punctuation when the string replaces the characters in the specified range.

```
- (NSString *)smartInsertAfterStringForString:(NSString *)aString
    replacingRange:(NSRange)charRange
```

Parameters

aString
The string that is replacing the characters in *charRange*.

charRange
The range of characters which *aString* is replacing.

Return Value
Any whitespace that needs to be added after *aString* to preserve proper spacing and punctuation when the characters in *charRange* are replaced by *aString*. If *aString* is `nil` or if smart insertion and deletion are disabled, this method returns `nil`.

Discussion
Don't invoke this method directly. Instead, use `smartInsertForString:replacingRange:beforeString:afterString:` (page 2953), which calls this method as part of its implementation.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextView.h`

smartInsertBeforeStringForString:replacingRange:

Returns any whitespace that needs to be added before the string to preserve proper spacing and punctuation when the string replaces the characters in the specified range.

```
- (NSString *)smartInsertBeforeStringForString:(NSString *)aString
    replacingRange:(NSRange)charRange
```

Parameters

aString
The string that is replacing the characters in *charRange*.

charRange

The range of characters which *aString* is replacing.

Return Value

Any whitespace that needs to be added before *aString* to preserve proper spacing and punctuation when the characters in *charRange* are replaced by *aString*. If *aString* is `nil` or if smart insertion and deletion are disabled, this method returns `nil`.

Discussion

Don't invoke this method directly. Instead, use `smartInsertForString:replacingRange:beforeString:afterString:` (page 2953), which calls this method as part of its implementation.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextView.h`

smartInsertDeleteEnabled

Returns whether the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation.

```
- (BOOL)smartInsertDeleteEnabled
```

Return Value

`YES` if the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation, `NO` if it inserts and deletes exactly what's selected.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `smartInsertForString:replacingRange:beforeString:afterString:` (page 2953)
- `smartDeleteRangeForProposedRange:` (page 2951)
- `setSmartInsertDeleteEnabled:` (page 2944)

Declared In

`NSTextView.h`

smartInsertForString:replacingRange:beforeString:afterString:

Determines whether whitespace needs to be added around the string to preserve proper spacing and punctuation when it replaces the characters in the specified range.

```
- (void)smartInsertForString:(NSString *)aString replacingRange:(NSRange)charRange
    beforeString:(NSString **)beforeString afterString:(NSString **)afterString
```

Parameters

aString

The string that is replacing the characters in *charRange*.

charRange
The range of characters which *aString* is replacing.

beforeString
On return, a pointer to the string with the characters that should be added before *aString*; `nil` if there are no characters to add, if *aString* is `nil`, or if smart insertion and deletion are disabled.

afterString
On return, a pointer to the string with the characters that should be added after *aString*; `nil` if there are no characters to add, if *aString* is `nil`, or if smart insertion and deletion are disabled.

Discussion
As part of its implementation, this method calls `smartInsertAfterStringForString:replacingRange:` (page 2952) and `smartInsertBeforeStringForString:replacingRange:` (page 2952). To change this method's behavior, override those two methods instead of this one.

`NSTextView` uses this method as necessary. You can also use it in implementing your own methods that insert text. To do so, invoke this method with the proper arguments, then insert *beforeString*, *aString*, and *afterString* in order over *charRange*.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `smartDeleteRangeForProposedRange:` (page 2951)
- `smartInsertDeleteEnabled` (page 2953)

Declared In
`NSTextView.h`

spellCheckerDocumentTag

Returns a tag identifying the text view's text as a document for the spell checker server.

```
- (NSInteger)spellCheckerDocumentTag
```

Return Value
A tag identifying the text view's text as a document for the spell checker server.

Discussion
The document tag is obtained by sending a `uniqueSpellDocumentTag` (page 2519) message to the spell server the first time this method is invoked for a particular group of text views. See the `NSSpellChecker` and `NSSpellServer` class specifications for more information on how this tag is used.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextView.h`

startSpeaking:

Speaks the selected text, or all text if no selection.

```
- (void)startSpeaking:(id)sender
```

Parameters

sender

The control sending the message; can be `nil`.

Availability
Available in Mac OS X v10.1 and later.

See Also
`- stopSpeaking:` (page 2955)

Declared In
`NSTextView.h`

stopSpeaking:

Stops the speaking of text.

```
- (void)stopSpeaking:(id)sender
```

Parameters

sender

The control sending the message; can be `nil`.

Availability
Available in Mac OS X v10.1 and later.

See Also
`- startSpeaking:` (page 2954)

Declared In
`NSTextView.h`

textContainer

Returns the receiver's text container.

```
- (NSTextContainer *)textContainer
```

Return Value
The receiver's text container.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTextContainer:` (page 2945)

Related Sample Code
CIAnnotation
LayoutManagerDemo
Sketch-112

TextSizingExample
TipWrapper

Declared In
NSTextView.h

textContainerInset

Returns the empty space the receiver leaves around its text container.

```
- (NSSize)textContainerInset
```

Return Value
The empty space the receiver leaves around its text container.

Availability
Available in Mac OS X v10.0 and later.

See Also
- textContainerOrigin (page 2956)
- invalidateTextContainerOrigin (page 2898)
- setTextContainerInset: (page 2946)

Declared In
NSTextView.h

textContainerOrigin

Returns the origin of the receiver's text container.

```
- (NSPoint)textContainerOrigin
```

Return Value
The origin of the receiver's text container, which is calculated from the receiver's bounds rectangle, container inset, and the container's used rect.

Availability
Available in Mac OS X v10.0 and later.

See Also
- invalidateTextContainerOrigin (page 2898)
- textContainerInset (page 2956)
- usedRectForTextContainer: (page 1524) (NSLayoutManager)

Related Sample Code
LayoutManagerDemo

Declared In
NSTextView.h

textStorage

Returns the receiver's text storage object.

```
- (NSTextStorage *)textStorage
```

Return Value
The receiver's text storage object.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BackgroundExporter
CIAnnotation
FunHouse
SBSetFinderComment
VertexPerformanceTest

Declared In
`NSTextView.h`

tightenKerning:

Decreases the space between glyphs in the receiver's selection, or for all glyphs if the receiver is a plain text view.

```
- (void)tightenKerning:(id)sender
```

Parameters

sender
The control that sent the message; may be `nil`.

Discussion
Kerning values are determined by the point size of the fonts in the selection.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `loosenKerning:` (page 2905)
- `useStandardKerning:` (page 2966)
- `turnOffKerning:` (page 2962)

Declared In
`NSTextView.h`

toggleAutomaticDashSubstitution:

Toggles the state of the automatic dash substitution.

```
- (void)toggleAutomaticDashSubstitution:(id)sender
```

Parameters

sender

The control sending the message. May be `nil`.

Discussion

Turning on automatic dash substitution enables automatic conversion of sequences of ASCII hyphen (`-`) characters to typographic dashes.

Availability

Available in Mac OS X v10.6 and later.

See Also

- `isAutomaticDashSubstitutionEnabled` (page 2899)
- `setAutomaticDashSubstitutionEnabled:` (page 2928)

Declared In

`NSTextView.h`

toggleAutomaticDataDetection:

Toggles the state of the automatic data detection.

```
- (void)toggleAutomaticDataDetection:(id)sender
```

Parameters

sender

The control sending the message. May be `nil`.

Discussion

Automatic data detection enables detection of dates, addresses, and phone numbers.

Availability

Available in Mac OS X v10.6 and later.

See Also

- `isAutomaticDataDetectionEnabled` (page 2899)
- `setAutomaticDataDetectionEnabled:` (page 2928)

Declared In

`NSTextView.h`

toggleAutomaticLinkDetection:

Changes the state of automatic link detection from enabled to disabled and vice versa.

```
- (void)toggleAutomaticLinkDetection:(id)sender
```

Parameters

sender

The control sending the message; may be `nil`.

Discussion

Automatic link detection causes strings representing URLs typed in the view to be automatically made into links to those URLs.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setAutomaticLinkDetectionEnabled:` (page 2929)
- `isAutomaticLinkDetectionEnabled` (page 2899)
`URLAtIndex:effectiveRange:` (page 270) (`NSAttributedString`)

Declared In
`NSTextView.h`

toggleAutomaticQuoteSubstitution:

Changes the state of automatic quotation mark substitution from enabled to disabled and vice versa.

```
- (void)toggleAutomaticQuoteSubstitution:(id)sender
```

Parameters
sender
The control sending the message; may be `nil`.

Discussion
Automatic quote substitution causes ASCII quotation marks and apostrophes to be automatically replaced, on a context-dependent basis, with more typographically accurate symbols.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `isAutomaticQuoteSubstitutionEnabled` (page 2900)
- `setAutomaticQuoteSubstitutionEnabled:` (page 2929)

Declared In
`NSTextView.h`

toggleAutomaticSpellingCorrection:

Toggles the state of the automatic spelling correction.

```
- (void)toggleAutomaticSpellingCorrection:(id)sender
```

Parameters
sender
The control sending the message. May be `nil`.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `isAutomaticSpellingCorrectionEnabled` (page 2900)
- `setAutomaticSpellingCorrectionEnabled:` (page 2929)

Declared In
NSTextView.h

toggleAutomaticTextReplacement:

Toggles the state of the automatic text replacement.

```
- (void)toggleAutomaticTextReplacement:(id)sender
```

Parameters

sender
The control sending the message. May be nil.

Discussion
Turning on automatic text replacement enables automatic substitution of a variety of static text items based on user preferences.

Availability
Available in Mac OS X v10.6 and later.

See Also
- isAutomaticTextReplacementEnabled (page 2901)
- setAutomaticTextReplacementEnabled: (page 2930)

Declared In
NSTextView.h

toggleBaseWritingDirection:

Changes the base writing direction of a paragraph between left-to-right and right-to-left. (Deprecated in Mac OS X v10.6.)

```
- (void)toggleBaseWritingDirection:(id)sender
```

Availability
Available in Mac OS X v10.3 and later.
Deprecated in Mac OS X v10.6.

Declared In
NSTextView.h

toggleContinuousSpellChecking:

Toggles whether continuous spell checking is enabled for the receiver.

```
- (void)toggleContinuousSpellChecking:(id)sender
```

Parameters

sender
The control sending the message; may be nil.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isContinuousSpellCheckingEnabled` (page 2902)
- `setContinuousSpellCheckingEnabled:` (page 2932)

Declared In
`NSTextView.h`

toggleGrammarChecking:

Changes the state of grammar checking from enabled to disabled and vice versa.

```
- (void)toggleGrammarChecking:(id)sender
```

Parameters

sender
The control sending the message; may be `nil`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setGrammarCheckingEnabled:` (page 2936)
- `isGrammarCheckingEnabled` (page 2903)

Declared In
`NSTextView.h`

toggleSmartInsertDelete:

Changes the state of smart insert and delete from enabled to disabled and vice versa.

```
- (void)toggleSmartInsertDelete:(id)sender
```

Parameters

sender
The control sending the message; may be `nil`.

Discussion
Controls whether the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `smartInsertDeleteEnabled` (page 2953)
- `setSmartInsertDeleteEnabled:` (page 2944)

Declared In
`NSTextView.h`

toggleTraditionalCharacterShape:

Toggles the `NSCharacterShapeAttributeName` attribute at the current selection.

```
- (void)toggleTraditionalCharacterShape:(id)sender
```

Parameters

sender

The control that sent the message; may be `nil`.

Discussion

The `NSCharacterShapeAttributeName` constant is defined in *NSAttributedString Application Kit Additions Reference*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextView.h`

turnOffKerning:

Sets the receiver to use nominal glyph spacing for the glyphs in its selection, or for all glyphs if the receiver is a plain text view.

```
- (void)turnOffKerning:(id)sender
```

Parameters

sender

The control that sent the message; may be `nil`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `useStandardKerning:` (page 2966)
- `loosenKerning:` (page 2905)
- `tightenKerning:` (page 2957)
- `isRichText` (page 2903)

Declared In

`NSTextView.h`

turnOffLigatures:

Sets the receiver to use only required ligatures when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.

```
- (void)turnOffLigatures:(id)sender
```

Parameters

sender

The control that sent the message; may be `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `useAllLigatures:` (page 2965)
- `isRichText` (page 2903)
- `useStandardLigatures:` (page 2967)

Declared In
`NSTextView.h`

typingAttributes

Returns the current typing attributes.

```
- (NSDictionary *)typingAttributes
```

Return Value
A dictionary of the current typing attributes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTypingAttributes:` (page 2947)

Declared In
`NSTextView.h`

updateDragTypeRegistration

Updates the acceptable drag types of all text views associated with the receiver's layout manager.

```
- (void)updateDragTypeRegistration
```

Discussion
If the receiver is editable and is a rich text view, causes all text views associated with the receiver's layout manager to register their acceptable drag types. If the text view isn't editable or isn't rich text, causes those text views to unregister their dragged types.

Subclasses can override this method to change the conditions for registering and unregistering drag types, whether as a group or individually based on the current state of the text view. They should invoke this method when that state changes to perform the necessary update.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `acceptableDragTypes` (page 2879)
- `registerForDraggedTypes:` (page 3201) (NSView)
- `unregisterDraggedTypes` (page 3239) (NSView)
- `isEditable` (page 2902)

- importsGraphics (page 2895)
- isRichText (page 2903)

Declared In
NSTextView.h

updateFontPanel

Updates the Font panel to contain the font attributes of the selection.

```
- (void)updateFontPanel
```

Discussion
Does nothing if the receiver doesn't use the Font panel. You should never need to invoke this method directly, but you can override it if needed to handle additional font attributes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- usesFontPanel (page 2966)

Declared In
NSTextView.h

updateInsertionPointStateAndRestartTimer:

Updates the insertion point's location and optionally restarts the blinking cursor timer.

```
- (void)updateInsertionPointStateAndRestartTimer:(BOOL)flag
```

Parameters
flag
YES to restart the blinking cursor timer, NO otherwise.

Discussion
This method is invoked automatically whenever the insertion point needs to be moved; you should never need to invoke it directly, but you can override it to modify insertion point behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
- shouldDrawInsertionPoint (page 2950)
- drawInsertionPointInRect:color:turnedOn: (page 2892)

Declared In
NSTextView.h

updateRuler

Updates the ruler view in the receiver's enclosing scroll view to reflect the selection's paragraph and marker attributes.

```
- (void)updateRuler
```

Discussion
Does nothing if the ruler isn't visible or if the receiver doesn't use the ruler. You should never need to invoke this method directly, but you can override this method if needed to handle additional ruler attributes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `usesRuler` (page 2966)

Declared In
`NSTextView.h`

useAllLigatures:

Sets the receiver to use all ligatures available for the fonts and languages used when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.

```
- (void)useAllLigatures:(id)sender
```

Parameters
sender
The control that sent the message; may be `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `turnOffLigatures:` (page 2962)
- `useStandardLigatures:` (page 2967)

Declared In
`NSTextView.h`

usesFindPanel

Returns whether the receiver allows for a find panel.

```
- (BOOL)usesFindPanel
```

Return Value
`YES` if the receiver allows the use of a find panel, `NO` otherwise.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setUsesFindPanel:` (page 2947)

Declared In
`NSTextView.h`

usesFontPanel

Returns whether the text views sharing the receiver's layout manager use the Font panel.

```
- (BOOL)usesFontPanel
```

Return Value
`YES` if the text views sharing the receiver's layout manager use the Font panel, `NO` otherwise.

Discussion
See `setUsesFontPanel:` (page 2947) and `rangeForUserCharacterAttributeChange` (page 2911) for the effect this method has on a text view's behavior.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextView.h`

usesRuler

Returns whether the text views sharing the receiver's layout manager use a ruler.

```
- (BOOL)usesRuler
```

Return Value
`YES` if the text views sharing the receiver's layout manager use a ruler, `NO` otherwise.

Discussion
See `setUsesRuler:` (page 2948) and `rangeForUserParagraphAttributeChange` (page 2912) for the effect this has on a text view's behavior. By default, text view objects use the ruler.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setUsesRuler:` (page 2948)

Declared In
`NSTextView.h`

useStandardKerning:

Set the receiver to use pair kerning data for the glyphs in its selection, or for all glyphs if the receiver is a plain text view.

```
- (void)useStandardKerning:(id)sender
```

Parameters

sender

The control that sent the message; may be `nil`.

Discussion

This data is taken from a font's AFM file

Availability

Available in Mac OS X v10.0 and later.

See Also

- `isRichText` (page 2903)
- `loosenKerning:` (page 2905)
- `tightenKerning:` (page 2957)
- `turnOffKerning:` (page 2962)

Declared In

`NSTextView.h`

useStandardLigatures:

Sets the receiver to use the standard ligatures available for the fonts and languages used when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.

```
- (void)useStandardLigatures:(id)sender
```

Parameters

sender

The control that sent the message; may be `nil`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `turnOffLigatures:` (page 2962)
- `useAllLigatures:` (page 2965)

Declared In

`NSTextView.h`

validRequestorForSendType:returnType:

Returns `self` if the text view can provide and accept the specified data types, or `nil` if it can't.

```
- (id)validRequestorForSendType:(NSString *)sendType returnType:(NSString
    *)returnType
```

Parameters

sendType

The type of data requested.

returnType
The type of data that will be returned.

Return Value
`self` if *sendType* specifies a type of data the text view can put on the pasteboard and *returnType* contains a type of data the text view can read from the pasteboard; otherwise `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- validRequestorForSendType:returnType:` (page 2204) (NSResponder)

Declared In
`NSTextView.h`

writablePasteboardTypes

Returns the pasteboard types that can be provided from the current selection.

```
- (NSArray *)writablePasteboardTypes
```

Return Value
An array of strings describing the types that can be written to the pasteboard immediately, or an array with no members if the text view has no text or no selection.

Discussion
Overriders can copy the result from super and add their own new types.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- readablePasteboardTypes` (page 2915)

Declared In
`NSTextView.h`

writeSelectionToPasteboard:type:

Writes the current selection to the specified pasteboard using the given type.

```
- (BOOL)writeSelectionToPasteboard:(NSPasteboard *)pboard type:(NSString *)type
```

Parameters
pboard
The pasteboard to write to.

type
The type of data to write.

Return Value
`YES` if the data was successfully written, `NO` otherwise.

Discussion
The complete set of data types being written to *pboard* should be declared before invoking this method.

This method should be invoked only from `writeSelectionToPasteboard:types:` (page 2969). You can override this method to add support for writing new types of data to the pasteboard. You should invoke `super`'s implementation of the method to handle any types of data your overridden version does not.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- readSelectionFromPasteboard:type:` (page 2916)

Declared In
`NSTextView.h`

writeSelectionToPasteboard:types:

Writes the current selection to the specified pasteboard under each given type.

```
- (BOOL)writeSelectionToPasteboard:(NSPasteboard *)pboard types:(NSArray *)types
```

Parameters

pboard
The pasteboard to write to.

types
An array of strings describing the types of data to write.

Return Value
`YES` if the data for any single type was successfully written, `NO` otherwise.

Discussion
This method declares the data types on *pboard* and then invokes `writeSelectionToPasteboard:type:` (page 2968) or the delegate method `textView:writeCell:atIndex:toPasteboard:type:` (page 3895) for each type in the *types* array.

You should not need to override this method. You might need to invoke this method if you are implementing a new type of pasteboard to handle services other than copy/paste or dragging.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextView.h`

Constants

NSSelectionGranularity

These constants specify how much the text view extends the selection when the user drags the mouse. They're used by `selectionGranularity` (page 2924), `setSelectionGranularity:` (page 2944), and `selectionRangeForProposedRange:granularity:` (page 2924):

```
typedef enum _NSSelectionGranularity {
    NSSelectByCharacter = 0,
    NSSelectByWord = 1,
    NSSelectByParagraph = 2
} NSSelectionGranularity;
```

Constants

`NSSelectByCharacter`
Extends the selection character by character.
Available in Mac OS X v10.0 and later.
Declared in `NSTextView.h`.

`NSSelectByWord`
Extends the selection word by word.
Available in Mac OS X v10.0 and later.
Declared in `NSTextView.h`.

`NSSelectByParagraph`
Extends the selection paragraph by paragraph.
Available in Mac OS X v10.0 and later.
Declared in `NSTextView.h`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextView.h`

NSSelectionAffinity

These constants specify the preferred direction of selection. They're used by `selectionAffinity` (page 2923) and `setSelectedRange:affinity:stillSelecting:` (page 2941).

```
typedef enum _NSSelectionAffinity {
    NSSelectionAffinityUpstream = 0,
    NSSelectionAffinityDownstream = 1
} NSSelectionAffinity;
```

Constants

`NSSelectionAffinityUpstream`
The selection is moving toward the top of the document.
Available in Mac OS X v10.0 and later.
Declared in `NSTextView.h`.

`NSSelectionAffinityDownstream`
The selection is moving toward the bottom of the document.

Available in Mac OS X v10.0 and later.

Declared in `NSTextView.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextView.h`

NSFindPanelAction

These constants define the tags for `performFindPanelAction:` (page 2910).

```
typedef enum {
    NSFindPanelActionShowFindPanel = 1,
    NSFindPanelActionNext = 2,
    NSFindPanelActionPrevious = 3,
    NSFindPanelActionReplaceAll = 4,
    NSFindPanelActionReplace = 5,
    NSFindPanelActionReplaceAndFind = 6,
    NSFindPanelActionSetFindString = 7,
    NSFindPanelActionReplaceAllInSelection = 8
} NSFindPanelAction;
```

Constants
`NSFindPanelActionShowFindPanel`
Displays the find panel.

Available in Mac OS X v10.3 and later.

Declared in `NSTextView.h`.

`NSFindPanelActionNext`
Finds the next instance of the queried text.

Available in Mac OS X v10.3 and later.

Declared in `NSTextView.h`.

`NSFindPanelActionPrevious`
Finds the previous instance of the queried text.

Available in Mac OS X v10.3 and later.

Declared in `NSTextView.h`.

`NSFindPanelActionReplaceAll`
Replaces all query instances within the text view.

Available in Mac OS X v10.3 and later.

Declared in `NSTextView.h`.

`NSFindPanelActionReplace`
Replaces a single query instance within the text view.

Available in Mac OS X v10.3 and later.

Declared in `NSTextView.h`.

`NSFindPanelActionReplaceAndFind`
Replaces a single query instance and finds the next.

Available in Mac OS X v10.3 and later.

Declared in `NSTextView.h`.

`NSFindPanelActionSetFindString`
Sets the query string to the current selection.

Available in Mac OS X v10.3 and later.

Declared in `NSTextView.h`.

`NSFindPanelActionReplaceAllInSelection`
Replaces all query instances within the selection.

Available in Mac OS X v10.3 and later.

Declared in `NSTextView.h`.

`NSFindPanelActionSelectAll`
Selects all query instances in the text view.

Available in Mac OS X v10.4 and later.

Declared in `NSTextView.h`.

`NSFindPanelActionSelectAllInSelection`
Selects all query instances within the selection.

Available in Mac OS X v10.4 and later.

Declared in `NSTextView.h`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSTextView.h`

Input Sources Locale Identifiers

Locale identifiers represent the input sources available.

```
NSString *NSAllRomanInputSourcesLocaleIdentifier
```

Constants
`NSAllRomanInputSourcesLocaleIdentifier`
A meta-locale identifier representing the set of Roman input sources available. You can pass `[NSArray arrayWithObject: NSAllRomanInputSourcesLocaleIdentifier]` to the `setAllowedInputSourceLocales:` (page 2926) method to restrict allowed input sources to Roman only.

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

Declared In
`NSTextView.h`

Find Panel Search Metadata

In addition to communicating search strings via the find pasteboard, the standard Find panel for `NSTextView` also communicates search option metadata, including case sensitivity and substring matching options. This metadata is stored in a property list as the `NSFindPanelSearchOptionsPboardType` (page 2973) value on the global find pasteboard. As such, third party applications may store additional keys in this property list to communicate additional metadata as desired to support the various search options common to many third-party applications' Find panels.

```
NSString *NSFindPanelSearchOptionsPboardType
NSString *NSFindPanelCaseInsensitiveSearch
NSString *NSFindPanelSubstringMatch
```

Constants

`NSFindPanelSearchOptionsPboardType`

Type for `NSFindPanel` metadata property list. Used with the `NSPasteBoard` method `propertyListForType:` (page 1895).

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

`NSFindPanelCaseInsensitiveSearch`

Boolean value specifying whether the search is case-insensitive. `YES` specifies a case-insensitive search; otherwise, `NO`.

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

`NSFindPanelSubstringMatch`

`NSNumber` object containing one of the values defined in "NSFindPanelSubstringMatchType" (page 2973).

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

Declared In

`NSTextView.h`

NSFindPanelSubstringMatchType

The type of substring matching used by the Find panel.

```
enum {
    NSFindPanelSubstringMatchTypeContains = 0,
    NSFindPanelSubstringMatchTypeStartsWith = 1,
    NSFindPanelSubstringMatchTypeFullWord = 2,
    NSFindPanelSubstringMatchTypeEndsWith = 3
};
typedef NSUInteger NSFindPanelSubstringMatchType;
```

Constants

`NSFindPanelSubstringMatchTypeContains`

Finds a word containing the search string.

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

`NSFindPanelSubstringMatchTypeStartsWith`
Finds a word starting with the search string.

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

`NSFindPanelSubstringMatchTypeFullWord`
Finds a word exactly matching the search string.

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

`NSFindPanelSubstringMatchTypeEndsWith`
Finds a word ending with the search string.

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

Declared In
`NSTextView.h`

Notifications

`NSTextView` posts the following notifications as well as those declared by its superclasses, particularly `NSText`. See the "Notifications" (page 2752) section in the `NSText` class specification for those other notifications.

NSTextViewDidChangeSelectionNotification

Posted when the selected range of characters changes.

`NSTextView` posts this notification whenever `setSelectedRange:affinity:stillSelecting:` (page 2941) is invoked, either directly or through the many methods (`mouseDown:` (page 2164), `selectAll:` (page 2194), and so on) that invoke it indirectly. When the user is selecting text, this notification is posted only once, at the end of the selection operation. The text view's delegate receives a `textViewDidChangeSelection:` (page 3895) message when this notification is posted.

The notification object is the notifying text view. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSOldSelectedCharacterRange"`	An `NSValue` object containing an `NSRange` structure with the originally selected range.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextView.h`

NSTextViewWillChangeNotifyingTextViewNotification

Posted when a new text view is established as the text view that sends notifications.

This notification allows observers to reregister themselves for the new text view. Methods such as `removeTextContainerAtIndex:` (page 1495), `textContainerChangedTextView:` (page 1518), and `insertTextContainer:atIndex:` (page 1480) cause this notification to be posted.

The notification object is the old notifying text view, or `nil`. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSOldNotifyingTextView"`	The old `NSTextView`, if one exists, otherwise `nil`.
`@"NSNewNotifyingTextView"`	The new `NSTextView`, if one exists, otherwise `nil`.

There's no delegate method associated with this notification. The text-handling system ensures that when a new text view replaces an old one as the notifying text view, the existing delegate becomes the delegate of the new text view, and the delegate is registered to receive text view notifications from the new notifying text view. All other observers are responsible for registering themselves on receiving this notification.

Availability
Available in Mac OS X v10.0 and later.

See Also
`removeObserver:` (`NSNotificationCenter`)
`addObserver:selector:name:object:` (`NSNotificationCenter`)

Declared In
`NSTextView.h`

NSTextViewDidChangeTypingAttributesNotification

Posted when there is a change in the typing attributes within a text view. This notification is posted, via the `textViewDidChangeTypingAttributes:` (page 3896) delegate method, whether or not text has changed as a result of the attribute change.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSTextView.h`

NSTokenField Class Reference

Inherits from	NSTextField : NSControl : NSView : NSResponder : NSObject
Conforms to	NSUserInterfaceValidations (NSTextField) NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSTokenField.h
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Token Field Programming Guide

Overview

`NSTokenField` is a subclass of `NSTextField` that provides tokenized editing similar to the address field in the Mail application.

`NSTokenField` uses an `NSTokenFieldCell` to implement much of the control's functionality. `NSTokenField` provides cover methods for most methods of `NSTokenFieldCell`, which invoke the corresponding cell method.

Notes: In Mac OS X v10.4 and earlier, represented objects associated with token fields had to conform to `NSCoding`. Starting with Mac OS X v10.5, they no longer need to.

In Mac OS X v10.4, `NSTokenField` trims whitespace around tokens but it does not trim whitespace in Mac OS X versions 10.5.0 and 10.5.1. In Mac OS X v10.5.2, you get whitespace-trimming behavior by either linking against the v10.4 binary or linking against the v10.5 binary and *not* implementing the `tokenField:representedObjectForEditingString:` (page 3909) method. If you do not want the whitespace-trimming behavior, link against the v10.5 binary and implement this method, returning the editing string if you have no represented object.

Tasks

Configuring the Token Style

`- setTokenStyle:` (page 2981)
Returns the token style of the receiver.

`- tokenStyle` (page 2982)
Returns the receiver's token style.

Configuring the Tokenizing Character Set

`- setTokenizingCharacterSet:` (page 2981)
Sets the recevier's tokenizing character set to *`characterSet`*.

`- tokenizingCharacterSet` (page 2981)
Returns the receiver's tokenizing character set.

`+ defaultTokenizingCharacterSet` (page 2979)
Returns the default tokenizing character set.

Configuring the Completion Delay

`- setCompletionDelay:` (page 2980)
Sets the receiver's completion delay.

`- completionDelay` (page 2979)
Returns the receiver's completion delay.

`+ defaultCompletionDelay` (page 2979)
Returns the default completion delay.

Getting and Setting the Delegate

`- delegate` (page 2980)
Returns the token field's delegate.

- setDelegate: (page 2980)
  Sets the token field's delegate

Class Methods

defaultCompletionDelay

Returns the default completion delay.

```
+ (NSTimeInterval)defaultCompletionDelay
```

Discussion
The default completion delay is 0.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTokenField.h

defaultTokenizingCharacterSet

Returns the default tokenizing character set.

```
+ (NSCharacterSet *)defaultTokenizingCharacterSet
```

Discussion
The default tokenizing character set is ",".

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTokenField.h

Instance Methods

completionDelay

Returns the receiver's completion delay.

```
- (NSTimeInterval)completionDelay
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setCompletionDelay: (page 2980)

+ defaultCompletionDelay (page 2979)

Declared In
NSTokenField.h

delegate

Returns the token field's delegate.

- (id < NSTokenFieldDelegate >)delegate

Return Value
The token field's delegate

Availability
Available in Mac OS X v10.6 and later.

See Also
- setDelegate: (page 2980)

Declared In
NSTokenField.h

setCompletionDelay:

Sets the receiver's completion delay.

- (void)setCompletionDelay:(NSTimeInterval)*delay*

Availability
Available in Mac OS X v10.4 and later.

See Also
- completionDelay (page 2979)

Declared In
NSTokenField.h

setDelegate:

Sets the token field's delegate

- (void)setDelegate:(id < NSTokenFieldDelegate >)*anObject*

Parameters
anObject
The delegate for the receiver. The delegate must conform to the NSTokenFieldDelegate Protocol protocol.

Availability
Available in Mac OS X v10.6 and later.

See Also
- delegate (page 2980)

Declared In
NSTokenField.h

setTokenizingCharacterSet:

Sets the recevier's tokenizing character set to *characterSet*.

```
- (void)setTokenizingCharacterSet:(NSCharacterSet *)characterSet
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- tokenizingCharacterSet (page 2981)

Declared In
NSTokenField.h

setTokenStyle:

Returns the token style of the receiver.

```
- (void)setTokenStyle:(NSTokenStyle)style
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- tokenStyle (page 2982)

Declared In
NSTokenField.h

tokenizingCharacterSet

Returns the receiver's tokenizing character set.

```
- (NSCharacterSet *)tokenizingCharacterSet
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setTokenizingCharacterSet: (page 2981)
+ defaultTokenizingCharacterSet (page 2979)

Declared In
NSTokenField.h

tokenStyle

Returns the receiver's token style.

```
- (NSTokenStyle)tokenStyle
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTokenStyle:` (page 2981)

Declared In
`NSTokenField.h`

NSTokenFieldCell Class Reference

Inherits from	NSTextFieldCell : NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSTokenFieldCell.h
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Token Field Programming Guide

Overview

`NSTokenFieldCell` is a subclass of `NSTextFieldCell` that provides tokenized editing of an array of objects similar to the address field in the Mail application. The objects may be strings or objects that can be represented as strings. A single token field cell can be presented in an `NSTokenField` control.

Tasks

Managing the Token Style

`- setTokenStyle:` (page 2987)
Sets the token style of the receiver.

`- tokenStyle` (page 2988)
Returns the receiver's token style.

Managing the Tokenizing Character Set

`+ defaultTokenizingCharacterSet` (page 2985)
Returns the default tokenizing character set.

`- setTokenizingCharacterSet:` (page 2986)
Sets the receiver's tokenizing character set to a given character set.

- `tokenizingCharacterSet` (page 2987)
  Returns the receiver's tokenizing character set.

Configuring the Completion Delay

- `setCompletionDelay:` (page 2986)
  Sets the receiver's completion delay to a given delay.
- `completionDelay` (page 2985)
  Returns the receiver's completion delay.
+ `defaultCompletionDelay` (page 2984)
  Returns the default completion delay.

Managing the Delegate

- `delegate` (page 2985)
  Returns the receiver's delegate.
- `setDelegate:` (page 2986)
  Sets the receiver's delegate.

Class Methods

defaultCompletionDelay

Returns the default completion delay.

```
+ (NSTimeInterval)defaultCompletionDelay
```

Return Value
The default completion delay.

Discussion
The default completion delay is `0`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `completionDelay` (page 2985)
- `setCompletionDelay:` (page 2986)

Declared In
`NSTokenFieldCell.h`

defaultTokenizingCharacterSet

Returns the default tokenizing character set.

```
+ (NSCharacterSet *)defaultTokenizingCharacterSet
```

Return Value
The default tokenizing character set.

Discussion
The default tokenizing character set contains the single character "`,`".

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTokenizingCharacterSet:` (page 2986)
- `tokenizingCharacterSet` (page 2987)

Declared In
`NSTokenFieldCell.h`

Instance Methods

completionDelay

Returns the receiver's completion delay.

```
- (NSTimeInterval)completionDelay
```

Return Value
The receiver's completion delay.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setCompletionDelay:` (page 2986)
+ `defaultCompletionDelay` (page 2984)

Declared In
`NSTokenFieldCell.h`

delegate

Returns the receiver's delegate.

```
- (id < NSTokenFieldCellDelegate >)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setDelegate:` (page 2986)

Declared In
`NSTokenFieldCell.h`

setCompletionDelay:

Sets the receiver's completion delay to a given delay.

```
- (void)setCompletionDelay:(NSTimeInterval)delay
```

Parameters

delay
The delay for the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `completionDelay` (page 2985)
+ `defaultCompletionDelay` (page 2984)

Declared In
`NSTokenFieldCell.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSTokenFieldCellDelegate >)anObject
```

Parameters

anObject
The delegate for the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `delegate` (page 2985)

Declared In
`NSTokenFieldCell.h`

setTokenizingCharacterSet:

Sets the receiver's tokenizing character set to a given character set.

```
- (void)setTokenizingCharacterSet:(NSCharacterSet *)characterSet
```

Parameters

characterSet

The tokenizing character set for the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- tokenizingCharacterSet (page 2987)
+ defaultTokenizingCharacterSet (page 2985)

Declared In
NSTokenFieldCell.h

setTokenStyle:

Sets the token style of the receiver.

```
- (void)setTokenStyle:(NSTokenStyle)style
```

Parameters

style

The token style for the receiver. The valid values are described in "NSTokenStyle" (page 2988).

Availability
Available in Mac OS X v10.4 and later.

See Also
- tokenStyle (page 2988)

Declared In
NSTokenFieldCell.h

tokenizingCharacterSet

Returns the receiver's tokenizing character set.

```
- (NSCharacterSet *)tokenizingCharacterSet
```

Return Value
The receiver's tokenizing character set.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setTokenizingCharacterSet: (page 2986)
+ defaultTokenizingCharacterSet (page 2985)

Declared In
NSTokenFieldCell.h

tokenStyle

Returns the receiver's token style.

```
- (NSTokenStyle)tokenStyle
```

Return Value
The receiver's token style. The valid values are described in "NSTokenStyle" (page 2988).

Availability
Available in Mac OS X v10.4 and later.

See Also
- setTokenStyle: (page 2987)

Declared In
NSTokenFieldCell.h

Constants

NSTokenStyle

The NSTokenStyle constants define how tokens are displayed and editable in the NSTokenFieldCell. These values are used by tokenStyle (page 2988), setTokenStyle: (page 2987) and the delegate method tokenFieldCell:styleForRepresentedObject: (page 3903).

```
enum {
    NSDefaultTokenStyle,
    NSPlainTextTokenStyle,
    NSRoundedTokenStyle
};
typedef NSUInteger NSTokenStyle;
```

Constants

NSDefaultTokenStyle
Style best used for keyword type tokens.
Available in Mac OS X v10.4 and later.
Declared in NSTokenFieldCell.h.

NSPlainTextTokenStyle
Style to use for data you want represented as plain-text and without any token background.
Available in Mac OS X v10.4 and later.
Declared in NSTokenFieldCell.h.

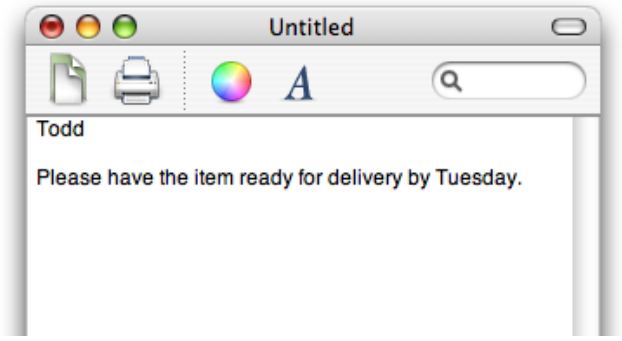
NSRoundedTokenStyle
Style best used for address type tokens.
Available in Mac OS X v10.4 and later.
Declared in NSTokenFieldCell.h.

NSToolbar Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSToolbar.h
Companion guide	Toolbar Programming Topics for Cocoa
Related sample code	GridCalendar iSpend Quartz Composer QCTV SimpleToolbar ToolbarSample

Overview

NSToolbar and `NSToolbarItem` provide the mechanism for a titled window to display a toolbar just below its title bar, as shown below:



Tasks

Creating an NSToolbar Object

- `initWithIdentifier:` (page 2995)
  Initializes a newly allocated toolbar with the specified identifier.

Toolbar Attributes

- `displayMode` (page 2994)
  Returns the receiver's display mode.
- `setDisplayMode:` (page 3000)
  Sets the receiver's display mode.
- `showsBaselineSeparator` (page 3002)
  Returns a Boolean value that indicates whether the toolbar shows the separator between the toolbar and the main window contents.
- `setShowsBaselineSeparator:` (page 3001)
  Sets whether the toolbar shows the separator between the toolbar and the main window contents.
- `allowsUserCustomization` (page 2992)
  Returns a Boolean value that indicates whether users are allowed to modify the toolbar.
- `setAllowsUserCustomization:` (page 2997)
  Sets whether users are allowed to modify the toolbar.
- `identifier` (page 2994)
  Returns the receiver's identifier.
- `items` (page 2996)
  Returns the receiver's current items, in order.
- `visibleItems` (page 3003)
  Returns the receiver's currently visible items.
- `sizeMode` (page 3002)
  Returns the receiver's size mode.
- `setSizeMode:` (page 3001)
  Sets the receiver's size mode.

Getting and Setting the Delegate

- `delegate` (page 2993)
  Returns the receiver's delegate.
- `setDelegate:` (page 2999)
  Sets the receiver's delegate.

Managing Items on the Toolbar

- `insertItemWithItemIdentifier:atIndex:` (page 2995)
  Inserts the specified item at the specified index.
- `removeItemAtIndex:` (page 2996)
  Removes the specified item.
- `setSelectedItemIdentifier:` (page 3000)
  Sets the receiver's selected item to the specified toolbar item.
- `selectedItemIdentifier` (page 2997)
  Returns the identifier of the receiver's currently selected item, or `nil` if there is no selection.

Displaying the Toolbar

- `isVisible` (page 2996)
  Returns a Boolean value that indicates whether the receiver is visible.
- `setVisible:` (page 3002)
  Sets whether the receiver is visible or hidden.

Toolbar Customization

- `runCustomizationPalette:` (page 2997)
  Runs the receiver's customization palette.
- `customizationPaletteIsRunning` (page 2993)
  Returns a Boolean value that indicates whether the receiver's customization palette is running (in use).

Autosaving the Configuration

- `autosavesConfiguration` (page 2992)
  Returns a Boolean value that indicates whether the receiver autosaves its configuration.
- `setAutosavesConfiguration:` (page 2998)
  Sets whether the receiver autosaves its configuration.
- `configurationDictionary` (page 2993)
  Returns the receiver's configuration as a dictionary.
- `setConfigurationFromDictionary:` (page 2999)
  Sets the receiver's configuration using *`configDict`*.

Validating Visible Items

- `validateVisibleItems` (page 3003)
  Called on window updates to validate the visible items.

Instance Methods

allowsUserCustomization

Returns a Boolean value that indicates whether users are allowed to modify the toolbar.

```
- (BOOL)allowsUserCustomization
```

Return Value
`YES` if users are allowed to modify the toolbar, `NO` otherwise. The default is `NO`.

Discussion
If the value is `NO`, then the Customize Toolbar… menu item is disabled and other modification is disabled. This attribute does not affect the user's ability to show or hide the toolbar.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAllowsUserCustomization:` (page 2997)
`- autosavesConfiguration` (page 2992)

Declared In
`NSToolbar.h`

autosavesConfiguration

Returns a Boolean value that indicates whether the receiver autosaves its configuration.

```
- (BOOL)autosavesConfiguration
```

Return Value
`YES` if the receiver autosaves its configuration, otherwise `NO`. The default is `NO`.

Discussion
When autosaving is enabled, the receiver will automatically write the toolbar settings to user defaults if the toolbar configuration changes. The toolbar's configuration is identified in user defaults by the toolbar identifier. If there are multiple toolbars active with the same identifier, they all share the same configuration.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAutosavesConfiguration:` (page 2998)
`- configurationDictionary` (page 2993)

Declared In
`NSToolbar.h`

configurationDictionary

Returns the receiver's configuration as a dictionary.

```
- (NSDictionary *)configurationDictionary
```

Return Value
A dictionary containing configuration information for the toolbar.

Discussion
Contains `displayMode`, `isVisible`, and a list of the item identifiers currently in the toolbar.

Special Considerations

Do not depend on any details of the normal contents of a configuration dictionary.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setConfigurationFromDictionary:` (page 2999)

Declared In
`NSToolbar.h`

customizationPaletteIsRunning

Returns a Boolean value that indicates whether the receiver's customization palette is running (in use).

```
- (BOOL)customizationPaletteIsRunning
```

Return Value
`YES` if the receiver's customization palette is running, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- runCustomizationPalette:` (page 2997)

Declared In
`NSToolbar.h`

delegate

Returns the receiver's delegate.

```
- (id < NSToolbarDelegate >)delegate
```

Return Value
The receiver's delegate.

Discussion
Every toolbar must have a delegate, which must implement the required delegate methods.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDelegate:` (page 2999)

Declared In
`NSToolbar.h`

displayMode

Returns the receiver's display mode.

```
- (NSToolbarDisplayMode)displayMode
```

Return Value
The receiver's display mode.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDisplayMode:` (page 3000)

Declared In
`NSToolbar.h`

identifier

Returns the receiver's identifier.

```
- (NSString *)identifier
```

Return Value
The receiver's identifier, a string used by the class to identify the kind of toolbar.

Discussion
Within the application all toolbars with the same identifier are synchronized to maintain the same state, including for example, the display mode and item order. The identifier is used as the autosave name for toolbars that save their configuration.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAutosavesConfiguration:`

Declared In
`NSToolbar.h`

initWithIdentifier:

Initializes a newly allocated toolbar with the specified identifier.

```
- (id)initWithIdentifier:(NSString *)identifier
```

Parameters

identifier

A string used by the class to identify the kind of the toolbar.

Return Value

The initialized toolbar object.

Discussion

identifier is never seen by users and should not be localized. See `identifier` (page 2994) for important information.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- identifier` (page 2994)

Declared In

`NSToolbar.h`

insertItemWithItemIdentifier:atIndex:

Inserts the specified item at the specified index.

```
- (void)insertItemWithItemIdentifier:(NSString *)itemIdentifier
    atIndex:(NSInteger)index
```

Parameters

itemIdentifier

The identifier of the item to insert.

index

The index at which to insert the item.

Discussion

If the toolbar needs a new instance, it will get it from `toolbar:itemForItemIdentifier:willBeInsertedIntoToolbar:` (page 3914). Typically, you should not call this method; you should let the user reconfigure the toolbar. See `identifier` (page 2994) for important information.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- removeItemAtIndex:` (page 2996)

Declared In

`NSToolbar.h`

isVisible

Returns a Boolean value that indicates whether the receiver is visible.

```
- (BOOL)isVisible
```

Return Value
`YES` if the receiver is visible, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setVisible:` (page 3002)

Declared In
`NSToolbar.h`

items

Returns the receiver's current items, in order.

```
- (NSArray *)items
```

Return Value
An array of the items in the toolbar.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- visibleItems` (page 3003)

Declared In
`NSToolbar.h`

removeItemAtIndex:

Removes the specified item.

```
- (void)removeItemAtIndex:(NSInteger)index
```

Parameters
index
The index of the item to remove.

Discussion
Typically, you should not call this method; you should let the user reconfigure the toolbar. See `identifier` (page 2994) for important information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `insertItemWithItemIdentifier:atIndex:` (page 2995)

Declared In
`NSToolbar.h`

runCustomizationPalette:

Runs the receiver's customization palette.

```
- (void)runCustomizationPalette:(id)sender
```

Parameters

sender
The control sending the message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `customizationPaletteIsRunning` (page 2993)

Declared In
`NSToolbar.h`

selectedItemIdentifier

Returns the identifier of the receiver's currently selected item, or `nil` if there is no selection.

```
- (NSString *)selectedItemIdentifier
```

Return Value
The identifier of the receiver's currently selected item, or `nil` if there is no selection.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setSelectedItemIdentifier:` (page 3000)

Declared In
`NSToolbar.h`

setAllowsUserCustomization:

Sets whether users are allowed to modify the toolbar.

```
- (void)setAllowsUserCustomization:(BOOL)allowsCustomization
```

Parameters

allowsCustomization
`YES` to allow users to modify the toolbar, `NO` otherwise.

Discussion
This value can be changed at any time. For instance, you may not want users to be able to customize the toolbar while some event is being processed. This attribute does not affect the user's ability to show or hide the toolbar.

If you set the toolbar to allow customization, be sure to also set the toolbar to autosave its configuration so the user's changes persist.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsUserCustomization` (page 2992)
- `setAutosavesConfiguration:` (page 2998)

Related Sample Code
iSpend
PDFKitLinker2
Quartz Composer QCTV
SimpleToolbar
ToolbarSample

Declared In
`NSToolbar.h`

setAutosavesConfiguration:

Sets whether the receiver autosaves its configuration.

```
- (void)setAutosavesConfiguration:(BOOL)flag
```

Parameters
flag
`YES` to indicate that the receiver should autosave its configuration, `NO` otherwise.

Discussion
Customizable toolbars should generally supporting autosaving. If you need to customize the saving behavior, you can use the `configurationDictionary` (page 2993) to access the settings that should be saved.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allowsUserCustomization` (page 2992)
- `autosavesConfiguration` (page 2992)

Related Sample Code
EnhancedDataBurn
iSpend
PDFKitLinker2
SimpleToolbar
ToolbarSample

Declared In
NSToolbar.h

setConfigurationFromDictionary:

Sets the receiver's configuration using *configDict*.

```
- (void)setConfigurationFromDictionary:(NSDictionary *)configDict
```

Parameters

configDict

A dictionary with the toolbar's configuration information. If you want to provide a custom dictionary, you should first get the receiver's current configuration dictionary, then create a modified copy, rather than trying to construct one yourself.

Discussion
This method immediately affects toolbars with the same identifier in all windows of your application.

Special Considerations

Do not depend on any details of the normal contents of a configuration dictionary.

Availability
Available in Mac OS X v10.0 and later.

See Also
- configurationDictionary (page 2993)

Declared In
NSToolbar.h

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id < NSToolbarDelegate >)delegate
```

Parameters

delegate

The new delegate object.

Discussion
Every toolbar must have a delegate, which must implement the required delegate methods.

Availability
Available in Mac OS X v10.0 and later.

See Also
- delegate (page 2993)

Related Sample Code
iSpend
PDFKitLinker2
Quartz Composer QCTV

SimpleToolbar
ToolbarSample

Declared In
NSToolbar.h

setDisplayMode:

Sets the receiver's display mode.

```
- (void)setDisplayMode:(NSToolbarDisplayMode)displayMode
```

Parameters
displayMode
The new display mode.

Availability
Available in Mac OS X v10.0 and later.

See Also
- displayMode (page 2994)

Related Sample Code
EnhancedDataBurn
iSpend
PDFKitLinker2
Quartz Composer QCTV
ToolbarSample

Declared In
NSToolbar.h

setSelectedItemIdentifier:

Sets the receiver's selected item to the specified toolbar item.

```
- (void)setSelectedItemIdentifier:(NSString *)itemIdentifier
```

Parameters
itemIdentifier
The identifier of the item to select. *itemIdentifier* may be any identifier returned by toolbarSelectableItemIdentifiers: (page 3916), even if it is not currently in the toolbar.

Discussion
Typically, a toolbar will manage the selection of items automatically. This method can be used to select identifiers of custom view items, or to force a selection change. See toolbarSelectableItemIdentifiers: (page 3916) for more details. If *itemIdentifier* is not recognized by the receiver, the current selected item identifier does not change.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `selectedItemIdentifier` (page 2997)
- `toolbarSelectableItemIdentifiers:` (page 3916) (NSToolbarDelegate)

Declared In
`NSToolbar.h`

setShowsBaselineSeparator:

Sets whether the toolbar shows the separator between the toolbar and the main window contents.

```
- (void)setShowsBaselineSeparator:(BOOL)flag
```

Parameters

flag

`YES` if the toolbar should show the separator between the toolbar and the main window contents, otherwise `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `showsBaselineSeparator` (page 3002)

Declared In
`NSToolbar.h`

setSizeMode:

Sets the receiver's size mode.

```
- (void)setSizeMode:(NSToolbarSizeMode)sizeMode
```

Parameters

sizeMode

The new size mode.

Discussion
If there is no icon of the given size for a toolbar item, the toolbar item creates one by scaling an icon of another size.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `sizeMode` (page 3002)

Declared In
`NSToolbar.h`

setVisible:

Sets whether the receiver is visible or hidden.

```
- (void)setVisible:(BOOL)shown
```

Parameters

shown

`YES` to indicate the receiver should be made visible, `NO` to indicate it should be hidden.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- isVisible` (page 2996)

Declared In
`NSToolbar.h`

showsBaselineSeparator

Returns a Boolean value that indicates whether the toolbar shows the separator between the toolbar and the main window contents.

```
- (BOOL)showsBaselineSeparator
```

Return Value
`YES` if the toolbar shows the separator between the toolbar and the main window contents, otherwise `NO`. The default is `YES`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setShowsBaselineSeparator:` (page 3001)

Declared In
`NSToolbar.h`

sizeMode

Returns the receiver's size mode.

```
- (NSToolbarSizeMode)sizeMode
```

Return Value
The receiver's size mode.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- setSizeMode:` (page 3001)

Declared In
NSToolbar.h

validateVisibleItems

Called on window updates to validate the visible items.

```
- (void)validateVisibleItems
```

Discussion
You typically use this method by overriding it in a subclass. The default implementation of this method iterates through the list of visible items, sending each a `validate` message. Override it and call `super` if you want to know when this method is called.

In Mac OS X v 10.6 and later toolbars no longer automatically validate for some events, including: `NSLeftMouseDragged` (page 1094), `NSRightMouseDragged` (page 1094), `NSOtherMouseDragged` (page 1094), `NSMouseEntered` (page 1094), `NSMouseExited` (page 1094), `NSScrollWheel` (page 1095), `NSCursorUpdate` (page 1094), `NSKeyDown` (page 1094). In addition, validation for `NSKeyUp` (page 1095) and `NSFlagsChanged` (page 1095) events is deferred with the timer restarting for every new deferrable event. So a sequence of key events will not trigger any validation at all, until either a pause of .85 seconds, or an event other than `NSKeyUp` (page 1095) or `NSFlagsChanged` (page 1095) is processed. This change was made as an optimization.

To trigger validation for a single toolbar manually, send the toolbar a `validateVisibleItems` (page 3003) message. To trigger validation for all toolbars, invoke NSApplication's `setWindowsNeedUpdate:` (page 172) passing `YES`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSToolbar.h

visibleItems

Returns the receiver's currently visible items.

```
- (NSArray *)visibleItems
```

Return Value
An array of the toolbar's visible items.

Discussion
Items in the overflow menu are not considered visible.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `items` (page 2996)

Declared In
NSToolbar.h

Constants

NSToolbarDisplayMode

These constants specify toolbar display modes and are used by `displayMode` (page 2994) and `setDisplayMode:` (page 3000).

```
enum {
    NSToolbarDisplayModeDefault,
    NSToolbarDisplayModeIconAndLabel,
    NSToolbarDisplayModeIconOnly,
    NSToolbarDisplayModeLabelOnly
};
typedef NSUInteger NSToolbarDisplayMode;
```

Constants

`NSToolbarDisplayModeDefault`
The default display mode.

Available in Mac OS X v10.0 and later.

Declared in `NSToolbar.h`.

`NSToolbarDisplayModeIconAndLabel`
The toolbar will display icons and labels.

Available in Mac OS X v10.0 and later.

Declared in `NSToolbar.h`.

`NSToolbarDisplayModeIconOnly`
The toolbar will display only icons.

Available in Mac OS X v10.0 and later.

Declared in `NSToolbar.h`.

`NSToolbarDisplayModeLabelOnly`
The toolbar will display only labels.

Available in Mac OS X v10.0 and later.

Declared in `NSToolbar.h`.

Declared In

`NSToolbar.h`

NSToolbarSizeMode

These constants specify toolbar display modes and are used by `sizeMode` (page 3002) and `setSizeMode:` (page 3001).

```
enum {
    NSToolbarSizeModeDefault,
    NSToolbarSizeModeRegular,
    NSToolbarSizeModeSmall
};
typedef NSUInteger NSToolbarSizeMode;
```

Constants

`NSToolbarSizeModeDefault`
The toolbar uses the system-defined default size, which is `NSToolbarSizeModeRegular`.
Available in Mac OS X v10.2 and later.
Declared in `NSToolbar.h`.

`NSToolbarSizeModeRegular`
The toolbar uses regular-sized controls and 32 by 32 pixel icons.
Available in Mac OS X v10.2 and later.
Declared in `NSToolbar.h`.

`NSToolbarSizeModeSmall`
The toolbar uses small-sized controls and 24 by 24 pixel icons.
Available in Mac OS X v10.2 and later.
Declared in `NSToolbar.h`.

Notifications

NSToolbarDidRemoveItemNotification

Posted after an item is removed from a toolbar. The notification item is the `NSToolbar` object that had an item removed from it. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"item"`	The `NSToolbarItem` object that was removed.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- toolbarDidRemoveItem:` (page 3916) (NSToolbarDelegate)

Declared In
`NSToolbar.h`

NSToolbarWillAddItemNotification

Posted before a new item is added to the toolbar. The notification item is the `NSToolbar` object having an item added to it. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"item"`	The `NSToolbarItem` object being added.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- toolbarWillAddItem:` (page 3917) (NSToolbarDelegate)

Declared In
`NSToolbar.h`

NSToolbarItem Class Reference

Inherits from	NSObject
Conforms to	NSValidatedUserInterfaceItem NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSToolbarItem.h
Companion guide	Toolbar Programming Topics for Cocoa
Related sample code	From A View to A Movie From A View to A Picture PDFKitLinker2 SimpleToolbar ToolbarSample

Overview

Each item in an NSToolbar is an instance of NSToolbarItem.

Adopted Protocols

NSCopying
- `copyWithZone:`

NSValidatedUserInterfaceItem
- `action` (page 3925)
- `tag` (page 3926)

Tasks

Creating a Toolbar Item

- `initWithItemIdentifier:` (page 3011)
  Initialize the receiver with a given identifier.

Managing Attributes

- `itemIdentifier` (page 3012)
  Returns the receiver's identifier.
- `toolbar` (page 3023)
  Returns the toolbar that is using the receiver.
- `label` (page 3013)
  Returns the receiver's label.
- `setLabel:` (page 3017)
  Sets the receiver's label that appears in the toolbar.
- `paletteLabel` (page 3014)
  Returns the label that appears when the receiver is in the customization palette.
- `setPaletteLabel:` (page 3019)
  Sets the receiver's label that appears when it is in the customization palette.
- `toolTip` (page 3023)
  Returns the tooltip used when the receiver is displayed in the toolbar.
- `setToolTip:` (page 3021)
  Sets the tooltip to be used when the receiver is displayed in the toolbar.
- `menuFormRepresentation` (page 3014)
  Returns the receiver's menu form representation.
- `setMenuFormRepresentation:` (page 3018)
  Sets the receiver's menu form.
- `tag` (page 3022)
  Returns the receiver's tag.
- `setTag:` (page 3020)
  Sets the receiver's tag.
- `target` (page 3023)
  Returns the receiver's target.
- `setTarget:` (page 3020)
  Sets the receiver's target.
- `action` (page 3010)
  Returns the receiver's action.
- `setAction:` (page 3015)
  Sets the receiver's action.

- `isEnabled` (page 3012)
  Returns a Boolean value that indicates whether the receiver is enabled.
- `setEnabled:` (page 3016)
  Sets the receiver's enabled flag.
- `image` (page 3011)
  Returns the image of the receiver.
- `setImage:` (page 3016)
  Sets the image for the receiver or of the view.
- `view` (page 3024)
  Returns the receiver's view.
- `setView:` (page 3021)
  Use this method to make the receiver into a view item.
- `minSize` (page 3014)
  Returns the receiver's minimum size.
- `setMinSize:` (page 3019)
  Sets the receiver's minimum size to a given size.
- `maxSize` (page 3013)
  Returns the receiver's maximum size.
- `setMaxSize:` (page 3018)
  Sets the receiver's maximum size to a given size.

Visibility Priority

- `visibilityPriority` (page 3025)
  Returns the receiver's visibility priority.
- `setVisibilityPriority:` (page 3022)
  Sets the receiver's visibility priority.

Validation

- `validate` (page 3024)
  This method is called by the receiver's toolbar during validation.
- `autovalidates` (page 3010)
  Returns a Boolean value that indicates whether the receiver is automatically validated by the toolbar.
- `setAutovalidates:` (page 3016)
  Sets the receiver's auto validation flag.

Controlling Duplicates

- `allowsDuplicatesInToolbar` (page 3010)
  Returns a Boolean value that indicates whether the receiver can be represented in the toolbar at more than one position.

Instance Methods

action

Returns the receiver's action.

```
- (SEL)action
```

Return Value
The receiver's action.

Discussion
For custom view items, this method sends `action` to the view if it responds and returns the result.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAction:` (page 3015)

Related Sample Code
EnhancedDataBurn
ToolbarSample

Declared In
`NSToolbarItem.h`

allowsDuplicatesInToolbar

Returns a Boolean value that indicates whether the receiver can be represented in the toolbar at more than one position.

```
- (BOOL)allowsDuplicatesInToolbar
```

Return Value
`YES` to allow dragging the receiver into the toolbar at more than one position, otherwise `NO`.

Discussion
You use this method by overriding it in a subclass to always return `YES`; typically, you wouldn't call it. By default, if an item with the same identifier is already in the toolbar, dragging it in again will effectively move it to the new position.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSToolbarItem.h`

autovalidates

Returns a Boolean value that indicates whether the receiver is automatically validated by the toolbar.

```
- (BOOL)autovalidates
```

Return Value
`YES` if the receiver is automatically validated by the toolbar, otherwise `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAutovalidates:` (page 3016)

Declared In
`NSToolbarItem.h`

image

Returns the image of the receiver.

```
- (NSImage *)image
```

Return Value
The image of the receiver.

Discussion
For an image item this method returns the result of the most recent `setImage:` (page 3016). For view items, this method calls `image` on the view if it responds and returns the result.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setImage:` (page 3016)

Related Sample Code
ToolbarSample

Declared In
`NSToolbarItem.h`

initWithItemIdentifier:

Initialize the receiver with a given identifier.

```
- (id)initWithItemIdentifier:(NSString *)itemIdentifier
```

Parameters
itemIdentifier
The identifier for the receiver. *itemIdentifier* is never seen by users and should not be localized.

Discussion
The identifier is used by the toolbar and its delegate to identify the kind of the toolbar item.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
iSpend
PDFKitLinker2
Quartz Composer QCTV
SimpleToolbar
ToolbarSample

Declared In
`NSToolbarItem.h`

isEnabled

Returns a Boolean value that indicates whether the receiver is enabled.

```
- (BOOL)isEnabled
```

Return Value
`YES` if the receiver is enabled, otherwise `NO`.

Discussion
For a view item, this method calls `isEnabled` on the view if it responds and returns the result.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setEnabled:` (page 3016)
- `view` (page 3024)

Declared In
`NSToolbarItem.h`

itemIdentifier

Returns the receiver's identifier.

```
- (NSString *)itemIdentifier
```

Return Value
The receiver's identifier, which was provided in the initializer.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `initWithItemIdentifier:` (page 3011)

Declared In
`NSToolbarItem.h`

label

Returns the receiver's label.

```
- (NSString *)label
```

Return Value
The receiver's label, which normally appears in the toolbar and in the overflow menu.

Discussion
For a discussion on labels, see "Setting a Toolbar Item's Representation".

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setLabel:` (page 3017)
- `menuFormRepresentation` (page 3014)

Related Sample Code
PDFKitLinker2
Quartz Composer QCTV
SimpleToolbar
ToolbarSample

Declared In
`NSToolbarItem.h`

maxSize

Returns the receiver's maximum size.

```
- (NSSize)maxSize
```

Return Value
The receiver's maximum size.

Discussion
See "Setting a Toolbar Item's Size" for a discussion on item sizes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMaxSize:` (page 3018)

Related Sample Code
Quartz Composer QCTV

Declared In
`NSToolbarItem.h`

menuFormRepresentation

Returns the receiver's menu form representation.

```
- (NSMenuItem *)menuFormRepresentation
```

Return Value
The receiver's menu form representation.

Discussion
By default, this method returns `nil`, even though there is a default menu form representation.

For a discussion on menu forms, see "Setting a Toolbar Item's Representation".

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMenuFormRepresentation:` (page 3018)

Related Sample Code
ToolbarSample

Declared In
`NSToolbarItem.h`

minSize

Returns the receiver's minimum size.

```
- (NSSize)minSize
```

Return Value
The receiver's minimum size.

Discussion
See "Setting a Toolbar Item's Size" for a discussion on item sizes.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setMinSize:` (page 3019)

Declared In
`NSToolbarItem.h`

paletteLabel

Returns the label that appears when the receiver is in the customization palette.

```
- (NSString *)paletteLabel
```

Return Value
The label that appears when the receiver is in the customization palette.

Discussion
An item must have a palette label if the customization palette is to be used, and for most items it is reasonable to set `paletteLabel` to be the same value as `label` (page 3013). One reason for `paletteLabel` to be different from `label` (page 3013) would be if it's more descriptive; another might be if there is no `label` (page 3013).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setPaletteLabel:` (page 3019)

Related Sample Code
ToolbarSample

Declared In
`NSToolbarItem.h`

setAction:

Sets the receiver's action.

```
- (void)setAction:(SEL)action
```

Parameters
action
The action for the receiver.

Discussion
For a custom view item, this method calls `setAction:` on the view if it responds.

See *Action Messages* for additional information on action messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- action` (page 3010)
`- setTarget:` (page 3020)

Related Sample Code
EnhancedDataBurn
GridCalendar
iSpend
PDFKitLinker2
SimpleToolbar

Declared In
`NSToolbarItem.h`

setAutovalidates:

Sets the receiver's auto validation flag.

```
- (void)setAutovalidates:(BOOL)resistance
```

Parameters

resistance

YES to set the receiver to be automatically validated by the toolbar, otherwise NO.

Discussion

By default NSToolbar automatically invokes the receiver's validate method on a regular basis. If your validate method is time consuming, you can disable auto validation on a per toolbar item basis.

Availability

Available in Mac OS X v10.4 and later.

See Also

- autovalidates (page 3010)

Declared In

NSToolbarItem.h

setEnabled:

Sets the receiver's enabled flag.

```
- (void)setEnabled:(BOOL)enabled
```

Parameters

enabled

YES to enable the receiver, otherwise NO.

Discussion

For a custom view item, this method calls setEnabled: on the view if it responds.

Availability

Available in Mac OS X v10.0 and later.

See Also

- isEnabled (page 3012)

Declared In

NSToolbarItem.h

setImage:

Sets the image for the receiver or of the view.

```
- (void)setImage:(NSImage *)image
```

Parameters

image

The image for the receiver, or of the view if it has already been set for the receiver.

Discussion
For a custom view item (one whose view has already been set), this method calls `setImage:` on the view if it responds. If *`image`* contains multiple representations, `NSToolbarItem` chooses the most appropriately sized representation when displaying.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `image` (page 3011)
- `view` (page 3024)

Related Sample Code
EnhancedDataBurn
GridCalendar
iSpend
PDFKitLinker2
SimpleToolbar

Declared In
`NSToolbarItem.h`

setLabel:

Sets the receiver's label that appears in the toolbar.

```
- (void)setLabel:(NSString *)label
```

Parameters
`label`
The receiver's label that appears in the toolbar. The length of the label should be appropriate and not too long. The label may be empty.

Discussion
The implication is that the toolbar will draw the label for the receiver, and a redraw is triggered by this method. The toolbar is in charge of the label area. For a discussion on labels, see "Setting a Toolbar Item's Representation".

Availability
Available in Mac OS X v10.0 and later.

See Also
- `label` (page 3013)
- `paletteLabel` (page 3014)

Related Sample Code
EnhancedDataBurn
GridCalendar
iSpend
PDFKitLinker2
Quartz Composer QCTV

Declared In
NSToolbarItem.h

setMaxSize:

Sets the receiver's maximum size to a given size.

```
- (void)setMaxSize:(NSSize)size
```

Parameters

size
The maximum size for the receiver.

Discussion
See "Setting a Toolbar Item's Size" for a discussion on item sizes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- maxSize (page 3013)

Related Sample Code
iSpend
PDFKitLinker2
Quartz Composer QCTV
SimpleToolbar
ToolbarSample

Declared In
NSToolbarItem.h

setMenuFormRepresentation:

Sets the receiver's menu form.

```
- (void)setMenuFormRepresentation:(NSMenuItem *)menuItem
```

Parameters

menuItem
The menu form for the receiver.

Discussion
For a discussion on menu forms see "Setting a Toolbar Item's Representation".

Availability
Available in Mac OS X v10.0 and later.

See Also
- menuFormRepresentation (page 3014)

Related Sample Code
PDFKitLinker2

SimpleToolbar
ToolbarSample

Declared In
NSToolbarItem.h

setMinSize:

Sets the receiver's minimum size to a given size.

```
- (void)setMinSize:(NSSize)size
```

Parameters

size
The minimum size for the receiver.

Discussion
See "Setting a Toolbar Item's Size" for a discussion on item sizes.

Availability
Available in Mac OS X v10.0 and later.

See Also
- minSize (page 3014)

Related Sample Code
iSpend
PDFKitLinker2
Quartz Composer QCTV
SimpleToolbar
ToolbarSample

Declared In
NSToolbarItem.h

setPaletteLabel:

Sets the receiver's label that appears when it is in the customization palette.

```
- (void)setPaletteLabel:(NSString *)paletteLabel
```

Parameters

paletteLabel
The label that appears when the receiver is in the customization palette.

Discussion
An item must have a palette label if the customization palette is to be used, and for most items it is reasonable to set paletteLabel (page 3014) to be the same value as label (page 3013). One reason for paletteLabel (page 3014) to be different from label (page 3013) would be if it's more descriptive; another might be if there is no label (page 3013).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `paletteLabel` (page 3014)
- `setLabel:` (page 3017)

Related Sample Code
EnhancedDataBurn
GridCalendar
iSpend
PDFKitLinker2
ToolbarSample

Declared In
`NSToolbarItem.h`

setTag:

Sets the receiver's tag.

```
- (void)setTag:(NSInteger)tag
```

Parameters
tag
The tag for the receiver.

Discussion
You can use the tag for your own custom purpose.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `tag` (page 3022)

Declared In
`NSToolbarItem.h`

setTarget:

Sets the receiver's target.

```
- (void)setTarget:(id)target
```

Parameters
target
The target for the receiver.

Discussion
If *target* is `nil`, the toolbar will call `action` on the first responder that implements it (see About the Responder Chain).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `target` (page 3023)
- `setAction:` (page 3015)
- `validateToolbarItem:` (page 3919) (`NSToolbarValidation`)

Related Sample Code
EnhancedDataBurn
iSpend
PDFKitLinker2
SimpleToolbar
ToolbarSample

Declared In
`NSToolbarItem.h`

setToolTip:

Sets the tooltip to be used when the receiver is displayed in the toolbar.

```
- (void)setToolTip:(NSString *)toolTip
```

Parameters

toolTip
A string representing the tooltip to be used when the receiver is displayed in the toolbar.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `toolTip` (page 3023)

Related Sample Code
EnhancedDataBurn
GridCalendar
iSpend
PDFKitLinker2
ToolbarSample

Declared In
`NSToolbarItem.h`

setView:

Use this method to make the receiver into a view item.

```
- (void)setView:(NSView *)view
```

Parameters

view
The view for the receiver. The view and all of its contents must conform to the `NSCoding` protocol if the toolbar supports customization.

Discussion
Note that many of the set/get methods are implemented by calls forwarded to *view*, if it responds to it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- view (page 3024)
- setMaxSize: (page 3018)
- setMinSize: (page 3019)

Related Sample Code
iSpend
PDFKitLinker2
Quartz Composer QCTV
SimpleToolbar
ToolbarSample

Declared In
NSToolbarItem.h

setVisibilityPriority:

Sets the receiver's visibility priority.

```
- (void)setVisibilityPriority:(NSInteger)visibilityPriority
```

Parameters
visibilityPriority
The visibility priority for the receiver. The values for *visibilityPriority* are described in Item Priority (page 3026).

Availability
Available in Mac OS X v10.4 and later.

See Also
- visibilityPriority (page 3025)

Declared In
NSToolbarItem.h

tag

Returns the receiver's tag.

```
- (NSInteger)tag
```

Return Value
The receiver's tag.

Discussion
You can use the tag for your own custom purpose.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTag:` (page 3020)

Declared In
`NSToolbarItem.h`

target

Returns the receiver's target.

```
- (id)target
```

Return Value
The receiver's target.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTarget:` (page 3020)

Related Sample Code
EnhancedDataBurn
SimpleToolbar
ToolbarSample

Declared In
`NSToolbarItem.h`

toolbar

Returns the toolbar that is using the receiver.

```
- (NSToolbar *)toolbar
```

Return Value
The toolbar that is using the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSToolbarItem.h`

toolTip

Returns the tooltip used when the receiver is displayed in the toolbar.

```
- (NSString *)toolTip
```

Return Value
The tooltip used when the receiver is displayed in the toolbar.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setToolTip:` (page 3021)

Related Sample Code
ToolbarSample

Declared In
`NSToolbarItem.h`

validate

This method is called by the receiver's toolbar during validation.

```
- (void)validate
```

Discussion
You may invoke this method directly if you have disabled automatic validation for an item—typically you do this for performance reasons if your validation code is slow. For further discussion, see "Validating Toolbar Items".

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setEnabled:` (page 3016)

Related Sample Code
SimpleToolbar

Declared In
`NSToolbarItem.h`

view

Returns the receiver's view.

```
- (NSView *)view
```

Return Value
The receiver's view.

Discussion
Note that many of the set/get methods are implemented by calls forwarded to the `NSView` object referenced by this attribute, if the object responds to it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setView:` (page 3021)

Related Sample Code
PDFKitLinker2
Quartz Composer QCTV
SimpleToolbar
ToolbarSample

Declared In
`NSToolbarItem.h`

visibilityPriority

Returns the receiver's visibility priority.

```
- (NSInteger)visibilityPriority
```

Return Value
The receiver's visibility priority. Possible values are described in `Item Priority` (page 3026).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setVisibilityPriority:` (page 3022)

Declared In
`NSToolbarItem.h`

Constants

Standard Identifiers

`NSToolbarItem` defines the following standard toolbar item identifiers.

```
NSString *NSToolbarSeparatorItemIdentifier;
NSString *NSToolbarSpaceItemIdentifier;
NSString *NSToolbarFlexibleSpaceItemIdentifier;
NSString *NSToolbarShowColorsItemIdentifier;
NSString *NSToolbarShowFontsItemIdentifier;
NSString *NSToolbarCustomizeToolbarItemIdentifier;
NSString *NSToolbarPrintItemIdentifier;
```

Constants
`NSToolbarSeparatorItemIdentifier`
The Separator item.

Available in Mac OS X v10.0 and later.

Declared in `NSToolbarItem.h`.

`NSToolbarSpaceItemIdentifier`
The Space item.

Available in Mac OS X v10.0 and later.

Declared in `NSToolbarItem.h`.

`NSToolbarFlexibleSpaceItemIdentifier`
The Flexible Space item.

Available in Mac OS X v10.0 and later.

Declared in `NSToolbarItem.h`.

`NSToolbarShowColorsItemIdentifier`
The Colors item. Shows the color panel.

Available in Mac OS X v10.0 and later.

Declared in `NSToolbarItem.h`.

`NSToolbarShowFontsItemIdentifier`
The Fonts item. Shows the font panel.

Available in Mac OS X v10.0 and later.

Declared in `NSToolbarItem.h`.

`NSToolbarCustomizeToolbarItemIdentifier`
The Customize item. Shows the customization palette.

Available in Mac OS X v10.0 and later.

Declared in `NSToolbarItem.h`.

`NSToolbarPrintItemIdentifier`
The Print item. Sends `printDocument:` to `firstResponder`, but you can change this in `toolbarWillAddItem:` if you need to do so.

Available in Mac OS X v10.0 and later.

Declared in `NSToolbarItem.h`.

Discussion
The following figure illustrates the items in the order in which they are described above.

Declared In
`NSToolbarItem.h`

Item Priority

When a toolbar does not have enough space to fit all its items, it must push some items into the overflow menu. These values allow you to suggest a priority for a toolbar item.

```
enum {
   NSToolbarItemVisibilityPriorityStandard = 0,
   NSToolbarItemVisibilityPriorityLow  = -1000,
   NSToolbarItemVisibilityPriorityHigh  = 1000,
   NSToolbarItemVisibilityPriorityUser  = 2000
};
```

Constants

`NSToolbarItemVisibilityPriorityStandard`
The default visibility priority.

Available in Mac OS X v10.4 and later.

Declared in `NSToolbarItem.h`.

`NSToolbarItemVisibilityPriorityLow`
Items with this priority will be the first items to be pushed to the overflow menu.

Available in Mac OS X v10.4 and later.

Declared in `NSToolbarItem.h`.

`NSToolbarItemVisibilityPriorityHigh`
Items with this priority are less inclined to be pushed to the overflow menu.

Available in Mac OS X v10.4 and later.

Declared in `NSToolbarItem.h`.

`NSToolbarItemVisibilityPriorityUser`
Items with this priority are the last to be pushed to the overflow menu. Only the user should set items to this priority.

Available in Mac OS X v10.4 and later.

Declared in `NSToolbarItem.h`.

Discussion

To suggest that an item always remain visible, give it a value greater than `NSToolbarItemVisibilityPriorityStandard`, but less than `NSToolbarItemVisibilityPriorityUser`. In configurable toolbars, users can control the priority of an item and the priority is autosaved by the NSToolbar. These values are used by the `setVisibilityPriority:` (page 3022) and `visibilityPriority` (page 3025) methods:

Declared In

`NSToolbarItem.h`

NSToolbarItemGroup Class Reference

Inherits from	NSToolbarItem : NSObject
Conforms to	NSValidatedUserInterfaceItem (NSToolbarItem) NSCopying (NSToolbarItem) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	NSToolbarItemGroup.h
Companion guide	Toolbar Programming Topics for Cocoa

Overview

`NSToolbarItemGroup` is a subclass of `NSToolbarItem` which contains subitems. The views and labels of the subitems are used, but the parent's attributes take precedence.

To configure an instance of `NSToolbarItemGroup`, you first create the individual toolbar items that will be the subitems:

```
NSToolbarItem *item1 = [[[NSToolbarItem alloc] initWithItemIdentifier:@"Item1"]
 autorelease];
NSToolbarItem *item2 = [[[NSToolbarItem alloc] initWithItemIdentifier:@"Item2"]
 autorelease];
[item1 setImage:[NSImage imageNamed:@"LeftArrow"]];
[item2 setImage:[NSImage imageNamed:@"RightArrow"]];
[item1 setLabel:@"Prev"];
[item2 setLabel:@"Next"];
```

and then put them in a grouped item:

```
NSToolbarItemGroup *group = [[[NSToolbarItemGroup alloc]
initWithItemIdentifier:@"GroupItem"] autorelease];
[group setSubitems:[NSArray arrayWithObjects:item1, item2, nil]];
```

In this configuration, you get two grouped items, and two labels. This differs from ordinary `NSToolbarItem` objects because they are attached—the user drags them together as a single item rather than separately.

If you set a label on the parent item:

```
[group setLabel:@"Navigate"];
```

you get two grouped items with one shared label.

If instead you set a view on the parent item, you get two labels with one shared view:

```
[group setView:someSegmentedControl];
```

Tasks

Working with Subitems

- `subitems` (page 3030)
  Returns the subitems for the receiver.
- `setSubitems:` (page 3030)
  Sets the subitems for the receiver.

Instance Methods

setSubitems:

Sets the subitems for the receiver.

```
- (void)setSubitems:(NSArray *)subitems
```

Parameters

subitems
An array of instances of `NSToolbarItem` objects that form the subitems for the receiver.

Discussion
You should call this method to set the subitems before returning the item to the toolbar. `NSToolbarItemGroup` objects cannot contain other `NSToolbarItemGroup` objects as subitems.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSToolbarItemGroup.h`

subitems

Returns the subitems for the receiver.

```
- (NSArray *)subitems
```

Return Value
The subitems for the receiver.

Discussion
By default, an `NSToolbarItemGroup` instance has an empty array of subitems.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSToolbarItemGroup.h`

NSTouch Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	NSTouch.h
Related sample code	LightTable

Overview

An instance of the `NSTouch` class is a snapshot of a particular touch at an instant in time.

A touch event is not persistent throughout the touch, new instances are created as the touch progresses. The identity property is used to follow a specific touch across its lifetime.

Touches do not have a corresponding screen location. The first touch of a touch collection is latched to the view underlying the cursor using the same hit detection as mouse events. Additional touches on the same device are also latched to the same view as any other touching touches. A touch remains latched to its view until the touch has either ended or is cancelled.

Tasks

Properties of This Touch

`identity` (page 3034) *property*
Use this property to track changes to a particular touch during the touch's life. (read-only)

`phase` (page 3036) *property*
The current phase of the touch. (read-only)

`normalizedPosition` (page 3035) *property*
The normalized position of the touch. (read-only)

`isResting` (page 3035) *property*
Returns whether the touch is a resting touch. (read-only)

Properties of Touch Device

`device` (page 3034) *property*
The digitizer that generated the touch. Useful to distinguish touches emanating from multiple-device scenario. (read-only)

`deviceSize` (page 3034) *property*
The range of the touch device in points (72ppi). (read-only)

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

device

The digitizer that generated the touch. Useful to distinguish touches emanating from multiple-device scenario. (read-only)

```
@property(readonly, retain) id device
```

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTouch.h`

deviceSize

The range of the touch device in points (72ppi). (read-only)

```
@property(readonly) NSSize deviceSize
```

Discussion
The lower-left corner of the surface is considered (0,0).

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTouch.h`

identity

Use this property to track changes to a particular touch during the touch's life. (read-only)

```
@property(readonly, retain) id<NSObject, NSCopying> identity
```

Discussion
While touch identities may be re-used, they are unique during the life of the touch, even when multiple devices are present.

Identity objects implement the `NSCopying` protocol so that they may be used as keys in an `NSDictionary`. Use `isEqual:` to compare two touch identities.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
LightTable

Declared In
`NSTouch.h`

isResting

Returns whether the touch is a resting touch. (read-only)

```
@property(readonly) BOOL isResting
```

Discussion
Resting touches occur when a user simply rests their thumb on the trackpad device.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTouch.h`

normalizedPosition

The normalized position of the touch. (read-only)

```
@property(readonly) NSPoint normalizedPosition
```

Discussion
The normalized position is a scaled value between (0.0) and (1.0,1.0), where (0.0,0.0) is the lower-left position on the touch device.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
LightTable

Declared In
`NSTouch.h`

phase

The current phase of the touch. (read-only)

```
@property(readonly) NSTouchPhase phase
```

Discussion
See "NSTouchPhase" (page 3036) for possible values.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSTouch.h

Constants

NSTouchPhase

The possible phases of a touch. These constants are used by `phase` (page 3036).

```
enum {
    NSTouchPhaseBegan           = 1U << 0,
    NSTouchPhaseMoved           = 1U << 1,
    NSTouchPhaseStationary      = 1U << 2,
    NSTouchPhaseEnded           = 1U << 3,
    NSTouchPhaseCancelled       = 1U << 4,
    NSTouchPhaseTouching        = NSTouchPhaseBegan | NSTouchPhaseMoved |
NSTouchPhaseStationary,
    NSTouchPhaseAny             = NSUIntegerMax
};
typedef NSUInteger NSTouchPhase;
```

Constants

`NSTouchPhaseBegan`
A finger touched the device. Or, a resting touch transitioned to an active touch and resting touches are not wanted by the view hierarchy.
Available in Mac OS X v10.6 and later.
Declared in `NSTouch.h`.

`NSTouchPhaseMoved`
A finger moved on the device.
Available in Mac OS X v10.6 and later.
Declared in `NSTouch.h`.

`NSTouchPhaseStationary`
A finger is touching the device, but hasn't moved since the previous event.
Available in Mac OS X v10.6 and later.
Declared in `NSTouch.h`.

`NSTouchPhaseEnded`

A finger was lifted from the screen. Or, an active touch transitioned to a resting touch and resting touches are not wanted by the view hierarchy.

Available in Mac OS X v10.6 and later.

Declared in `NSTouch.h`.

`NSTouchPhaseCancelled`

The system cancelled tracking for the touch, as when (for example) the window associated with the touch resigns key or is deactivated.

Available in Mac OS X v10.6 and later.

Declared in `NSTouch.h`.

`NSTouchPhaseTouching`

Matches the `NSTouchPhaseBegan` (page 3036), `NSTouchPhaseMoved` (page 3036), or `NSTouchPhaseStationary` (page 3036) phases of a touch.

Available in Mac OS X v10.6 and later.

Declared in `NSTouch.h`.

`NSTouchPhaseAny`

Matches any phase of a touch.

Available in Mac OS X v10.6 and later.

Declared in `NSTouch.h`.

NSTrackingArea Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSTrackingArea.h
Availability	Available in Mac OS X v10.5 and later.
Companion guide	Cocoa Event-Handling Guide
Related sample code	BasicCocoaAnimations MenuItemView PhotoSearch TrackIt

Overview

An `NSTrackingArea` object defines a region of view that generates mouse-tracking and cursor-update events when the mouse is over that region.

When creating a tracking-area object, you specify a rectangle (in the view's coordinate system), an owning object, and one or more options, along with (optionally) a dictionary of data. Once it's created, you add the tracking-area object to a view using the `addTrackingArea:` (page 3142) method. Depending on the options specified, the owner of the tracking area receives `mouseEntered:` (page 2165), `mouseExited:` (page 2165), `mouseMoved:` (page 2166), and `cursorUpdate:` (page 2147) messages when the mouse cursor enters, moves within, and leaves the tracking area. Currently the tracking area is restricted to rectangles.

An `NSTrackingArea` object belongs to its view rather than to its window. Consequently, you can add and remove tracking rectangles without needing to worry if the view has been added to a window. In addition, this design makes it possible for the Application Kit to compute the geometry of tracking areas automatically when a view moves and, in some cases, when a view changes size.

With `NSTrackingArea`, you can configure the scope of activity for mouse tracking. There are four options:

- The tracking area is active only when the view is first responder.
- The tracking area is active when the view is in the key window.
- The tracking area is active when the application is active.

- The tracking area is active always (even when the application is inactive).

Other options for `NSTrackingArea` objects include specifying that the tracking area should be synchronized with the visible rectangle of the view (`visibleRect` (page 3245)) and for generating `mouseEntered:` and `mouseExited:` events when the mouse is dragged.

Other `NSView` methods related to `NSTrackingArea` objects (in addition to `addTrackingArea:`) include `removeTrackingArea:` (page 3204) and `updateTrackingAreas` (page 3239). Views can override the latter method to recompute and replace their `NSTrackingArea` objects in certain situations, such as a change in the size of the `visibleRect`.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

NSCopying
- `copyWithZone:`

Tasks

Initializing the Tracking-Area Object

- `initWithRect:options:owner:userInfo:` (page 3041)
  Initializes and returns an object defining a region of a view to receive mouse-tracking events, mouse-moved events, cursor-update events, or possibly all these events.

Getting Object Attributes

- `options` (page 3042)
  Returns the options specified for the receiver.
- `owner` (page 3042)
  Returns the object owning the receiver, which is the recipient of mouse-tracking, mouse-movement, and cursor-update messages.
- `rect` (page 3042)
  Returns the rectangle defining the area encompassed by the receiver.
- `userInfo` (page 3043)
  Returns the dictionary containing the data associated with the receiver when it was created.

Instance Methods

initWithRect:options:owner:userInfo:

Initializes and returns an object defining a region of a view to receive mouse-tracking events, mouse-moved events, cursor-update events, or possibly all these events.

```
- (id)initWithRect:(NSRect)rect options:(NSTrackingAreaOptions)options
    owner:(id)owner userInfo:(NSDictionary *)userInfo
```

Parameters

rect

A rectangle that defines a region of a target view, in the view's coordinate system, for tracking events related to mouse tracking and cursor updating. The specified rectangle should not exceed the view's bounds rectangle.

options

One or more constants that specify the type of tracking area, the situations when the area is active, and special behaviors of the tracking area. See the description of `NSTrackingAreaOptions` (page 3043) and related constants for details. You must specify one or more options for the initialized object, in particular the type of tracking area; zero is not a valid value.

owner

The object to receive the requested mouse-tracking, mouse-moved, or cursor-update messages. It does not necessarily have to be the view associated with the created `NSTrackingArea` object, but should be an object capable of responding to the `NSResponder` methods `mouseEntered:` (page 2165), `mouseExited:` (page 2165), `mouseMoved:` (page 2166), and `cursorUpdate:` (page 2147).

userInfo

A dictionary containing arbitrary data for each mouse-entered, mouse-exited, and cursor-update event. When handling such an event you can obtain the dictionary by sending `userData` (page 1089) to the `NSEvent` object. (The dictionary is not available for mouse-moved events.) This parameter may be `nil`.

Return Value

The newly-initialized tracking area object.

Discussion

After creating and initializing an `NSTrackingArea` object with this method, you must add it to a target view using the `addTrackingArea:` (page 3142) method. When changes in the view require changes in the geometry of its tracking areas, the Application Kit invokes `updateTrackingAreas` (page 3239). The view should implement this method to replace the current `NSTrackingArea` object with one with a recomputed area.

Special Considerations

Beginning with Mac OS X v10.5, the `initWithRect:options:owner:userInfo:`, along with the `addTrackingArea:` (page 3142) method of `NSView`, replace the `NSView` method `addTrackingRect:owner:userData:assumeInside:` (page 3142).

Availability

Available in Mac OS X v10.5 and later.

See Also

- `options` (page 3042)
- `owner` (page 3042)

- `rect` (page 3042)
- `userInfo` (page 3043)

Related Sample Code
BasicCocoaAnimations
MenuItemView
PhotoSearch
TrackIt

Declared In
`NSTrackingArea.h`

options

Returns the options specified for the receiver.

```
- (NSTrackingAreaOptions)options
```

Discussion
The options for an `NSTrackingArea` object are specified when the object is created. To determine if a particular option is in effect, perform a bitwise-AND operation with an `NSTrackingAreaOptions` (page 3043) constant and the value returned from this method, for example:

```
if ([trackingAreaObj options] & NSTrackingInVisibleRect != 0) {
    // do something appropriate
}
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTrackingArea.h`

owner

Returns the object owning the receiver, which is the recipient of mouse-tracking, mouse-movement, and cursor-update messages.

```
- (id)owner
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTrackingArea.h`

rect

Returns the rectangle defining the area encompassed by the receiver.

```
- (NSRect)rect
```

Discussion
The rectangle is specified in the local coordinate system of the associated view. if the `NSTrackingInVisibleRect` (page 3045) option is specified, the receiver is automatically synchronized with changes in the view's visible area (`visibleRect` (page 3245)) and the value returned from this method is ignored.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTrackingArea.h`

userInfo

Returns the dictionary containing the data associated with the receiver when it was created.

```
- (NSDictionary *)userInfo
```

Discussion
Returns `nil` if no data was specified when the receiver was initialized. You can obtain this dictionary per event in each `mouseEntered:` (page 2165) and `mouseExited:` (page 2165) method by querying the passed-in `NSEvent` object with [[event trackingArea] `userData` (page 1089)].

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTrackingArea.h`

Constants

NSTrackingAreaOptions

The data type defined for the constants specified in the *`options`* parameter of `initWithRect:options:owner:userInfo:` (page 3041). These constants are described below; you may specify multiple constants by performing a bitwise-OR operation with them.

```
typedef NSUInteger NSTrackingAreaOptions;
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTrackingArea.h`

The following constants specify the type of the tracking area defined by an `NSTrackingArea` object. They request the type of messages the owning object should receive.

Constant	Description
`NSTrackingMouseEnteredAndExited`	The owner of the tracking area receives `mouseEntered:` (page 2165) when the mouse cursor enters the area and `mouseExited:` (page 2165) events when the mouse leaves the area. Available in Mac OS X v10.5 and later. Declared in `NSTrackingArea.h`.
`NSTrackingMouseMoved`	The owner of the tracking area receives `mouseMoved:` (page 2166) messages while the mouse cursor is within the area. Available in Mac OS X v10.5 and later. Declared in `NSTrackingArea.h`.
`NSTrackingCursorUpdate`	The owner of the tracking area receives `cursorUpdate:` (page 2147) messages when the mouse cursor enters the area; when the mouse leaves the area, the cursor is appropriately reset. Available in Mac OS X v10.5 and later. Declared in `NSTrackingArea.h`.

The following constants specify when the tracking area defined by an `NSTrackingArea` object is active. The owner receives all requested messages—which can include `mouseEntered:` (page 2165), `mouseExited:` (page 2165), `mouseMoved:` (page 2166), and `cursorUpdate:` (page 2147)—unless otherwise noted.

Constant	Description
`NSTrackingActiveWhenFirstResponder`	The owner receives messages when the view is the first responder. Available in Mac OS X v10.5 and later. Declared in `NSTrackingArea.h`.
`NSTrackingActiveInKeyWindow`	The owner receives messages when the view is in the key window. Available in Mac OS X v10.5 and later. Declared in `NSTrackingArea.h`.
`NSTrackingActiveInActiveApp`	The owner receives messages when the application is active. Available in Mac OS X v10.5 and later. Declared in `NSTrackingArea.h`.
`NSTrackingActiveAlways`	The owner receives messages regardless of first-responder status, window status, or application status. The `cursorUpdate:` (page 2147) message is *not* sent when the `NSTrackingCursorUpdate` (page 3044) option is specified along with this constant. Available in Mac OS X v10.5 and later. Declared in `NSTrackingArea.h`.

The following constants specify various behaviors of the tracking defined by an `NSTrackingArea` object.

Constant	Description
`NSTrackingAssumeInside`	The first event is generated when the cursor leaves the tracking area, regardless if the cursor is inside the area when the `NSTrackingArea` is added to a view. If this option is not specified, the first event is generated when the cursor leaves the tracking area if the cursor is initially inside the area, or when the cursor enters the area if the cursor is initially outside it. Generally, you do not want to request this behavior. Available in Mac OS X v10.5 and later. Declared in `NSTrackingArea.h`.
`NSTrackingInVisibleRect`	Mouse tracking occurs only in the visible rectangle of the view—in other words, that region of the tracking rectangle that is unobscured. Otherwise, the entire tracking area is active regardless of overlapping views. The `NSTrackingArea` object is automatically synchronized with changes in the view's visible area (`visibleRect` (page 3245)) and the value returned from `rect` (page 3042) is ignored. Available in Mac OS X v10.5 and later. Declared in `NSTrackingArea.h`.
`NSTrackingEnabledDuringMouseDrag`	The owner receives `NSMouseEntered` (page 1094) events when the mouse cursor is dragged into the tracking area. If this option is not specified, the owner receives mouse-entered events when the mouse is moved (no buttons pressed) into the tracking area and on `NSLeftMouseUp` (page 1093) events after a mouse drag. `NSMouseExited` (page 1094) and `NSMouseEntered` (page 1094) events are paired so their delivery is indirectly affected. That is, if a `NSMouseEntered` event is generated and the mouse cursor subsequently moves out of the tracking area, a `NSMouseExited` event is generated regardless if the mouse is moved or dragged, independent of this constant. Available in Mac OS X v10.5 and later. Declared in `NSTrackingArea.h`.

NSTreeController Class Reference

Inherits from	NSObjectController : NSController : NSObject
Conforms to	NSCoding (NSController) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	AppKit/NSTreeController.h
Companion guide	Cocoa Bindings Programming Topics
Related sample code	AbstractTree CoreRecipes ImageApp QTMetadataEditor SourceView

Overview

The `NSTreeController` is a bindings compatible controller that manages a tree of objects. It provides selection and sort management. Its primary purpose is to act as the controller when binding NSOutlineView and NSBrowser instances to a hierarchical collection of objects. The root content object of the tree can be a single object, or an array of objects.

An `NSTreeController` requires that you describe how the tree of objects is traversed by specifying the key-path for child objects specified by `setChildrenKeyPath:` (page 3064). All child objects for the tree must be key-value coding compliant for the same child key-path. If necessary, you should implement accessor methods in your model classes, that map the child key name to the appropriate class-specific method name.

Child objects can implement a count method (specified to the tree controller using `setCountKeyPath:` (page 3065)) that, if provided, returns the number of child objects available. Your model objects are expected to update the value of the count key-path in a key-value observing compliant method. You can also, optionally, provide a leaf key-path using `setLeafKeyPath:` (page 3065) that specifies a key in your model object that returns `YES` if the object is a leaf node, and `NO` if it is not. Changes to the leaf node value of the child object should be made in a key-value observing compliant manner. Providing the leaf node key-path prevents the `NSTreeController` from having to determine if a child object is a leaf node by examining the child object and as a result improves performance.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

NSCopying
- `copyWithZone:`

Tasks

Managing Sort Descriptors

- `setSortDescriptors:` (page 3067)
  Sets the sort descriptors used to arrange the receiver's contents.
- `sortDescriptors` (page 3067)
  Returns an array containing the sort descriptors used to arrange the receiver's content.

Setting the Content

- `setContent:` (page 3064)
  Sets the receiver's content.
- `content` (page 3055)
  Returns the receiver's content object.

Arranging Objects

- `arrangedObjects` (page 3052)
  Returns the tree controller's sorted content objects.
- `rearrangeObjects` (page 3060)
  Use this method to trigger reordering of the receiver's content.

Getting the Current Selection

- `setSelectionIndexPath:` (page 3066)
  Sets the receiver's current selection.
- `selectionIndexPath` (page 3062)
  Returns the index path of the first selected object.
- `setSelectionIndexPaths:` (page 3066)
  Sets the receiver's current selection to the specified index paths.

- `selectionIndexPaths` (page 3062)
  Returns an array containing the index paths of the currently selected objects.
- `selectedObjects` (page 3062)
  Returns an array containing the currently selected objects.
- `selectedNodes` (page 3061)
  Returns an array of the receiver's selected tree nodes.

Managing Selections

- `setSelectsInsertedObjects:` (page 3067)
  Sets whether the receiver will automatically select objects as they are inserted.
- `selectsInsertedObjects` (page 3063)
  Returns whether the receiver selects inserted objects automatically.
- `addSelectionIndexPaths:` (page 3052)
  Adds the objects at the specified *`indexPaths`* in the receiver's content to the current selection.
- `removeSelectionIndexPaths:` (page 3061)
  Removes the objects at the specified indexPaths from the receiver's current selection, returning `YES` if the selection was changed.
- `setAvoidsEmptySelection:` (page 3064)
  Sets whether the receiver will attempt to avoid an empty selection.
- `avoidsEmptySelection` (page 3053)
  Returns whether the receiver requires that the content array attempt to maintain a selection at all times.
- `setPreservesSelection:` (page 3065)
  Sets whether the receiver will attempt to preserve selection when the content changes.
- `preservesSelection` (page 3059)
  Returns whether the receiver will attempt to preserve the current selection when the content changes.
- `setAlwaysUsesMultipleValuesMarker:` (page 3063)
  Sets whether the receiver always returns the multiple values marker when multiple objects are selected, even if they have the same value.
- `alwaysUsesMultipleValuesMarker` (page 3052)
  Returns whether the receiver always returns the multiple values marker when multiple objects are selected, even if the selected items have the same value.

Adding, Inserting and Removing Objects

- `add:` (page 3051)
  Adds an object to the receiver after the current selection.
- `addChild:` (page 3051)
  Adds a child object to the currently selected item.
- `canAddChild` (page 3053)
  Returns `YES` if a child object can be added to the receiver's content.
- `canInsert` (page 3054)
  Returns `YES` if an object can be inserted into the receiver's content.

- `canInsertChild` (page 3054)
  Returns `YES` if a child object can be inserted into the receiver's content.
- `insert:` (page 3056)
  Creates a new object of the class specified by `objectClass` and inserts it into the receiver's content.
- `insertChild:` (page 3057)
  Creates a new object of the class specified by `objectClass` and inserts it into the receiver's content as a child of the current selection.
- `insertObject:atArrangedObjectIndexPath:` (page 3057)
  Inserts *`object`* into the receiver's arranged objects array at the location specified by *`indexPath`*, and adds it to the receiver's content.
- `insertObjects:atArrangedObjectIndexPaths:` (page 3057)
  Inserts *`objects`* into the receiver's arranged objects array at the locations specified in *`indexPaths`*, and adds them to the receiver's content.
- `remove:` (page 3060)
  Removes the receiver's selected objects from the content.
- `removeObjectAtArrangedObjectIndexPath:` (page 3060)
  Removes the object at the specified *`indexPath`* in the receiver's arranged objects from the receiver's content.
- `removeObjectsAtArrangedObjectIndexPaths:` (page 3061)
  Removes the objects at the specified *`indexPaths`* in the receiver's arranged objects from the receiver's content.
- `moveNode:toIndexPath:` (page 3058)
  Moves the specified tree node to the new index path.
- `moveNodes:toIndexPath:` (page 3059)
  Moves the specified tree nodes to the new index path.

Specifying Model Attributes

- `setChildrenKeyPath:` (page 3064)
  Sets the key path used by the receiver to access child objects to *`key`*.
- `childrenKeyPath` (page 3054)
  Returns the key path used to find the children in the receiver's objects.
- `setCountKeyPath:` (page 3065)
  Sets the key path used by the receiver to determine the number of objects at a node to *`key`*.
- `childrenKeyPathForNode:` (page 3055)
  Returns the key path used to find the children in the specified tree node.
- `countKeyPath` (page 3055)
  Returns the key path used to find the number of children for a node.
- `countKeyPathForNode:` (page 3056)
  Returns the key path that provides the number of children for a specified node.
- `setLeafKeyPath:` (page 3065)
  Sets the key path used by the receiver to determine if an object is a leaf node to *`key`*.
- `leafKeyPath` (page 3058)
  Returns the key path used by the receiver to determine if a node is a leaf key.

- leafKeyPathForNode: (page 3058)
  Returns the key path that specifies whether the node is a leaf node.

Instance Methods

add:

Adds an object to the receiver after the current selection.

```
- (void)add:(id)sender
```

Discussion
The *sender* is typically the object that invoked this method.

If the receiver is in object mode `newObject` is called and the returned object is added to the collection. If the receiver is in entity mode a new object is created that is appropriate as specified by the entity, and `newObject` is not used.

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism can provide feedback as a sheet.

Availability
Available in Mac OS X v10.4 and later.

See Also
- remove: (page 3060)

Declared In
`NSTreeController.h`

addChild:

Adds a child object to the currently selected item.

```
- (void)addChild:(id)sender
```

Discussion
The *sender* is typically the object that invoked this method.

If the receiver is in object mode `newObject` is called and the returned object is added as a child. If the receiver is in entity mode a new object is created that is appropriate for the relationship as specified by the entity, and `newObject` is not used.

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism can provide feedback as a sheet.

Availability
Available in Mac OS X v10.4 and later.

See Also
- remove: (page 3060)

Declared In
NSTreeController.h

addSelectionIndexPaths:

Adds the objects at the specified *indexPaths* in the receiver's content to the current selection.

```
- (BOOL)addSelectionIndexPaths:(NSArray *)indexPaths
```

Discussion
Attempting to change the selection may cause a commitEditing message which fails, thus denying the selection change.

Availability
Available in Mac OS X v10.4 and later.

See Also
- removeSelectionIndexPaths: (page 3061)

Declared In
NSTreeController.h

alwaysUsesMultipleValuesMarker

Returns whether the receiver always returns the multiple values marker when multiple objects are selected, even if the selected items have the same value.

```
- (BOOL)alwaysUsesMultipleValuesMarker
```

Discussion
The default is NO.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setAlwaysUsesMultipleValuesMarker: (page 3063)

Declared In
NSTreeController.h

arrangedObjects

Returns the tree controller's sorted content objects.

```
- (id)arrangedObjects
```

Return Value
A proxy root tree note containing the receiver's sorted content objects. The proxy object responds to `childNodes` (page 3071) and `descendantNodeAtIndexPath:` (page 3071) messages.

Discussion
This property is observable using key-value observing.

Special Considerations

Prior to Mac OS X v10.5 this method returned an opaque root node object representing all the currently displayed objects. That return value was only suitable for use with Cocoa bindings, and no assumption should be made about what methods that opaque object supports.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- rearrangeObjects` (page 3060)

Declared In
`NSTreeController.h`

avoidsEmptySelection

Returns whether the receiver requires that the content array attempt to maintain a selection at all times.

```
- (BOOL)avoidsEmptySelection
```

Discussion
The default is `YES`.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAvoidsEmptySelection:` (page 3064)

Declared In
`NSTreeController.h`

canAddChild

Returns `YES` if a child object can be added to the receiver's content.

```
- (BOOL)canAddChild
```

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
- canInsertChild (page 3054)

Declared In
NSTreeController.h

canInsert

Returns YES if an object can be inserted into the receiver's content.

```
- (BOOL)canInsert
```

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
- canInsertChild (page 3054)

Declared In
NSTreeController.h

canInsertChild

Returns YES if a child object can be inserted into the receiver's content.

```
- (BOOL)canInsertChild
```

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
- insertChild: (page 3057)

Declared In
NSTreeController.h

childrenKeyPath

Returns the key path used to find the children in the receiver's objects.

```
- (NSString *)childrenKeyPath
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- setChildrenKeyPath: (page 3064)

Declared In
NSTreeController.h

childrenKeyPathForNode:

Returns the key path used to find the children in the specified tree node.

```
- (NSString *)childrenKeyPathForNode:(NSTreeNode *)node
```

Parameters

node

A tree node in the receiver.

Return Value
A string containing the key path in *node* that provides the child nodes.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTreeController.h

content

Returns the receiver's content object.

```
- (id)content
```

Return Value
The receiver's content object.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setContent: (page 3064)

Declared In
NSTreeController.h

countKeyPath

Returns the key path used to find the number of children for a node.

```
- (NSString *)countKeyPath
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setCountKeyPath:` (page 3065)

Declared In
`NSTreeController.h`

countKeyPathForNode:

Returns the key path that provides the number of children for a specified node.

```
- (NSString *)countKeyPathForNode:(NSTreeNode *)node
```

Parameters

node

A tree node in the receiver.

Return Value
A string containing the key path in *node* that provides the number of children.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTreeController.h`

insert:

Creates a new object of the class specified by `objectClass` and inserts it into the receiver's content.

```
- (void)insert:(id)sender
```

Discussion
The *sender* is typically the object that invoked this method.

If the receiver is in object mode `newObject` is called and the returned object is inserted into the collection. If the receiver is in entity mode a new object is created that is appropriate as specified by the entity, and `newObject` is not used.

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism can provide feedback as a sheet.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `add:` (page 3051)

Declared In
`NSTreeController.h`

insertChild:

Creates a new object of the class specified by `objectClass` and inserts it into the receiver's content as a child of the current selection.

```
- (void)insertChild:(id)sender
```

Discussion
The *sender* is typically the object that invoked this method.

If the receiver is in object mode `newObject` is called and the returned object is inserted as a child. If the receiver is in entity mode a new object is created that is appropriate for the relationship as specified by the entity, and `newObject` is not used.

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism can provide feedback as a sheet.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `add:` (page 3051)

Declared In
`NSTreeController.h`

insertObject:atArrangedObjectIndexPath:

Inserts *object* into the receiver's arranged objects array at the location specified by *indexPath*, and adds it to the receiver's content.

```
- (void)insertObject:(id)object atArrangedObjectIndexPath:(NSIndexPath *)indexPath
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `insertObjects:atArrangedObjectIndexPaths:` (page 3057)

Related Sample Code
SourceView

Declared In
`NSTreeController.h`

insertObjects:atArrangedObjectIndexPaths:

Inserts *objects* into the receiver's arranged objects array at the locations specified in *indexPaths*, and adds them to the receiver's content.

```
- (void)insertObjects:(NSArray *)objects atArrangedObjectIndexPaths:(NSArray
    *)indexPaths
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `removeObjectAtArrangedObjectIndexPath:` (page 3060)

Declared In
`NSTreeController.h`

leafKeyPath

Returns the key path used by the receiver to determine if a node is a leaf key.

```
- (NSString *)leafKeyPath
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setLeafKeyPath:` (page 3065)

Declared In
`NSTreeController.h`

leafKeyPathForNode:

Returns the key path that specifies whether the node is a leaf node.

```
- (NSString *)leafKeyPathForNode:(NSTreeNode *)node
```

Parameters
node
A tree node in the receiver.

Return Value
A string containing the key path in *node* that specifies that the node is a leaf node.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTreeController.h`

moveNode:toIndexPath:

Moves the specified tree node to the new index path.

```
- (void)moveNode:(NSTreeNode *)node toIndexPath:(NSIndexPath *)indexPath
```

Parameters
node
A tree node.

indexPath
An index path specifying the new position in the receiver's content.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
SourceView

Declared In
NSTreeController.h

moveNodes:toIndexPath:

Moves the specified tree nodes to the new index path.

```
- (void)moveNodes:(NSArray *)nodes toIndexPath:(NSIndexPath *)startingIndexPath
```

Parameters
nodes
An array of tree nodes.

startingIndexPath
An index path specifying the starting position to move the tree nodes to in the receiver's content.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTreeController.h

preservesSelection

Returns whether the receiver will attempt to preserve the current selection when the content changes.

```
- (BOOL)preservesSelection
```

Discussion
The default is YES.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setPreservesSelection: (page 3065)

Declared In
NSTreeController.h

rearrangeObjects

Use this method to trigger reordering of the receiver's content.

```
- (void)rearrangeObjects
```

Discussion
Subclasses should invoke this method if any parameter that affects the arranged objects changes.

Availability
Available in Mac OS X v10.4 and later.

See Also
- arrangedObjects (page 3052)

Declared In
NSTreeController.h

remove:

Removes the receiver's selected objects from the content.

```
- (void)remove:(id)sender
```

Discussion
The *sender* is typically the object that invoked this method.

Special Considerations

Beginning with Mac OS X v10.4 the result of this method is deferred until the next iteration of the runloop so that the error presentation mechanism can provide feedback as a sheet.

Availability
Available in Mac OS X v10.4 and later.

See Also
- add: (page 3051)

Related Sample Code
SourceView

Declared In
NSTreeController.h

removeObjectAtArrangedObjectIndexPath:

Removes the object at the specified *indexPath* in the receiver's arranged objects from the receiver's content.

```
- (void)removeObjectAtArrangedObjectIndexPath:(NSIndexPath *)indexPath
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- removeObjectsAtArrangedObjectIndexPaths: (page 3061)

Declared In
NSTreeController.h

removeObjectsAtArrangedObjectIndexPaths:

Removes the objects at the specified *indexPaths* in the receiver's arranged objects from the receiver's content.

```
- (void)removeObjectsAtArrangedObjectIndexPaths:(NSArray *)indexPaths
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- removeObjectAtArrangedObjectIndexPath: (page 3060)

Declared In
NSTreeController.h

removeSelectionIndexPaths:

Removes the objects at the specified indexPaths from the receiver's current selection, returning YES if the selection was changed.

```
- (BOOL)removeSelectionIndexPaths:(NSArray *)indexPaths
```

Discussion
Attempting to change the selection may cause a commitEditing message which fails, thus denying the selection change.

Availability
Available in Mac OS X v10.4 and later.

See Also
- addSelectionIndexPaths: (page 3052)

Declared In
NSTreeController.h

selectedNodes

Returns an array of the receiver's selected tree nodes.

```
- (NSArray *)selectedNodes
```

Return Value
An array containing the receiver's selected tree nodes

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTreeController.h

selectedObjects

Returns an array containing the currently selected objects.

```
- (NSArray *)selectedObjects
```

Return Value
An array containing the currently selected objects in the tree controller content.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
QTMetadataEditor

Declared In
NSTreeController.h

selectionIndexPath

Returns the index path of the first selected object.

```
- (NSIndexPath *)selectionIndexPath
```

Return Value
Returns an index path of the first object in the tree controller's selection. Returns `nil` if there is no selection.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `selectionIndexPaths` (page 3062)

Related Sample Code
SourceView

Declared In
NSTreeController.h

selectionIndexPaths

Returns an array containing the index paths of the currently selected objects.

```
- (NSArray *)selectionIndexPaths
```

Return Value
An array containing `NSIndexPath` objects for each of the selected objects in the tree controller's content.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- selectionIndexPath` (page 3062)

Declared In
`NSTreeController.h`

selectsInsertedObjects

Returns whether the receiver selects inserted objects automatically.

```
- (BOOL)selectsInsertedObjects
```

Discussion
The default is `YES`.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setSelectsInsertedObjects:` (page 3067)

Declared In
`NSTreeController.h`

setAlwaysUsesMultipleValuesMarker:

Sets whether the receiver always returns the multiple values marker when multiple objects are selected, even if they have the same value.

```
- (void)setAlwaysUsesMultipleValuesMarker:(BOOL)flag
```

Discussion
Setting *flag* to `YES` can increase performance if your application doesn't allow editing multiple values. The default is `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- alwaysUsesMultipleValuesMarker` (page 3052)

Declared In
NSTreeController.h

setAvoidsEmptySelection:

Sets whether the receiver will attempt to avoid an empty selection.

```
- (void)setAvoidsEmptySelection:(BOOL)flag
```

Discussion
If *flag* is YES then the receiver will maintain a selection unless there are no objects in the content. The default is YES.

Availability
Available in Mac OS X v10.4 and later.

See Also
- avoidsEmptySelection (page 3053)

Declared In
NSTreeController.h

setChildrenKeyPath:

Sets the key path used by the receiver to access child objects to *key*.

```
- (void)setChildrenKeyPath:(NSString *)key
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- childrenKeyPath (page 3054)

Declared In
NSTreeController.h

setContent:

Sets the receiver's content.

```
- (void)setContent:(id)content
```

Parameters
content
The content. The content can be an array of objects, or a single root object.

Availability
Available in Mac OS X v10.4 and later.

See Also
- content (page 3055)

Related Sample Code
ImageApp

Declared In
`NSTreeController.h`

setCountKeyPath:

Sets the key path used by the receiver to determine the number of objects at a node to *key*.

```
- (void)setCountKeyPath:(NSString *)key
```

Discussion
Specifying this key path, if the data is available in the model object, can increase performance, but disables insert and remove functionality.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `countKeyPath` (page 3055)

Declared In
`NSTreeController.h`

setLeafKeyPath:

Sets the key path used by the receiver to determine if an object is a leaf node to *key*.

```
- (void)setLeafKeyPath:(NSString *)key
```

Discussion
Specifying this key path is optional. If the receiver is able to determine that a node is a leaf node, it can disable inserting or adding children to those nodes.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `leafKeyPath` (page 3058)

Declared In
`NSTreeController.h`

setPreservesSelection:

Sets whether the receiver will attempt to preserve selection when the content changes.

```
- (void)setPreservesSelection:(BOOL)flag
```

Discussion
If *flag* is `YES` then the selection will be preserved, if possible. The default is `YES`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `preservesSelection` (page 3059)

Declared In
`NSTreeController.h`

setSelectionIndexPath:

Sets the receiver's current selection.

```
- (BOOL)setSelectionIndexPath:(NSIndexPath *)indexPath
```

Parameters
indexPath
The proposed new selection.

Return Value
Return `YES` if the selection has changed, otherwise `NO`.

Discussion
Attempting to change the selection may cause a `commitEditing` message which fails, thus denying the selection change.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `selectionIndexPaths` (page 3062)

Declared In
`NSTreeController.h`

setSelectionIndexPaths:

Sets the receiver's current selection to the specified index paths.

```
- (BOOL)setSelectionIndexPaths:(NSArray *)indexPaths
```

Parameters
indexPaths
An array of NSIndexpath objects specifying the selected objects.

Return Value
Return `YES` if the selection has changed, otherwise `NO`.

Discussion
Attempting to change the selection may cause a `commitEditing` message which fails, thus denying the selection change.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setSelectionIndexPath:` (page 3066)

Declared In
`NSTreeController.h`

setSelectsInsertedObjects:

Sets whether the receiver will automatically select objects as they are inserted.

```
- (void)setSelectsInsertedObjects:(BOOL)flag
```

Discussion
If *flag* is `YES` then items will be selected upon insertion. The default is `YES`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `selectsInsertedObjects` (page 3063)

Declared In
`NSTreeController.h`

setSortDescriptors:

Sets the sort descriptors used to arrange the receiver's contents.

```
- (void)setSortDescriptors:(NSArray *)sortDescriptors
```

Parameters

sortDescriptors
An array of `NSSortDescriptor` objects. Passing `nil` causes the contents to be arranged in their natural order.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `sortDescriptors` (page 3067)

Declared In
`NSTreeController.h`

sortDescriptors

Returns an array containing the sort descriptors used to arrange the receiver's content.

```
- (NSArray *)sortDescriptors
```

Return Value
Returns an array containing the tree controller's sort descriptors. Returns `nil` if the receiver has no sort descriptors configured.

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setSortDescriptors:` (page 3067)

Declared In
`NSTreeController.h`

NSTreeNode Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSTreeNode.h
Companion guide	Cocoa Bindings Programming Topics
Related sample code	AbstractTree DragNDropOutlineView SourceView

Overview

`NSTreeNode` simplifies the creation and management of trees of objects. Each tree node represents a model object. A tree node with `nil` as its parent node is considered the root of the tree.

Tasks

Creating Tree Nodes

`+ treeNodeWithRepresentedObject:` (page 3070)
Creates and returns a tree node that represents the specified object.

`- initWithRepresentedObject:` (page 3072)
Initializes a newly allocated tree node that represents the specified object.

Getting Information About a Node

`- representedObject` (page 3073)
Returns the object the tree node represents.

`- indexPath` (page 3071)
Returns the position of the receiver relative to its root parent.

- `isLeaf` (page 3072)
  Returns whether the receiver is a leaf node.
- `childNodes` (page 3071)
  Returns an array containing receiver's child nodes.
- `mutableChildNodes` (page 3072)
  Returns a mutable array that provides read-write access to the receiver's child nodes.
- `descendantNodeAtIndexPath:` (page 3071)
  Returns the receiver's descendent at the specified index path.
- `parentNode` (page 3073)
  Returns the receiver's parent node.

Sorting the Subtree

- `sortWithSortDescriptors:recursively:` (page 3074)
  Sorts the receiver's subtree using the values of the represented objects with the specified sort descriptors.

Class Methods

treeNodeWithRepresentedObject:

Creates and returns a tree node that represents the specified object.

```
+ (id)treeNodeWithRepresentedObject:(id)modelObject
```

Parameters

modelObject
  The object the tree node represents.

Return Value
An initialized tree node that represents *modelObject*.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
DragNDropOutlineView

Declared In
`NSTreeNode.h`

Instance Methods

childNodes

Returns an array containing receiver's child nodes.

```
- (NSArray *)childNodes
```

Return Value
An array containing the receiver's child nodes.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
DragNDropOutlineView

Declared In
`NSTreeNode.h`

descendantNodeAtIndexPath:

Returns the receiver's descendent at the specified index path.

```
- (NSTreeNode *)descendantNodeAtIndexPath:(NSIndexPath *)indexPath
```

Parameters

indexPath
An index path specifying a descendent of the receiver.

Return Value
A tree node, or `nil` if the node does not exist.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTreeNode.h`

indexPath

Returns the position of the receiver relative to its root parent.

```
- (NSIndexPath *)indexPath
```

Return Value
An index path that represents the receiver's position relative to the tree's root node.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
AbstractTree
SourceView

Declared In
`NSTreeNode.h`

initWithRepresentedObject:

Initializes a newly allocated tree node that represents the specified object.

```
- (id)initWithRepresentedObject:(id)modelObject
```

Parameters
modelObject
The object the tree node represents.

Return Value
An initialized tree node that represents *modelObject*.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTreeNode.h`

isLeaf

Returns whether the receiver is a leaf node.

```
- (BOOL)isLeaf
```

Return Value
`YES` if the receiver is a leaf node (has no child nodes), otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTreeNode.h`

mutableChildNodes

Returns a mutable array that provides read-write access to the receiver's child nodes.

```
- (NSMutableArray *)mutableChildNodes
```

Return Value
A mutable array that provides read-write access to the receiver's child nodes.

Discussion
Nodes that are inserted into this array have their parent nodes set to the receiver. Nodes that are removed from this array automatically have their parent node set to `nil`. The array that is returned is observable using key-value observing.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
DragNDropOutlineView

Declared In
`NSTreeNode.h`

parentNode

Returns the receiver's parent node.

```
- (NSTreeNode *)parentNode
```

Return Value
The receiver's parent node.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
AbstractTree
DragNDropOutlineView
SourceView

Declared In
`NSTreeNode.h`

representedObject

Returns the object the tree node represents.

```
- (id)representedObject
```

Return Value
The object the tree node represents.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
DragNDropOutlineView

Declared In
`NSTreeNode.h`

sortWithSortDescriptors:recursively:

Sorts the receiver's subtree using the values of the represented objects with the specified sort descriptors.

```
- (void)sortWithSortDescriptors:(NSArray *)sortDescriptors
    recursively:(BOOL)recursively
```

Parameters

sortDescriptors

Array of sort descriptors specifying how to sort the represented objects.

recursively

A Boolean that specifies whether the child nodes should be sorted recursively.

Discussion

All the represented objects in the child nodes must be key-value coding compliant for the keys specified in the sort descriptors.

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

DragNDropOutlineView

Declared In

`NSTreeNode.h`

NSTypesetter Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTypesetter.h
Companion guides	Text System Overview Text Layout Programming Guide

Overview

`NSLayoutManager` uses concrete subclasses of this abstract class, `NSTypesetter`, to perform line layout, which includes word wrapping, hyphenation, and line breaking in either vertical or horizontal rectangles. By default, the text system uses the concrete subclass `NSATSTypesetter`.

Subclassing Notes

`NSTypesetter` provides concrete subclasses with default implementation interfacing with the Cocoa text system. By subclassing `NSTypesetter`, an application can override the `layoutParagraphAtPoint:` (page 3094) method to integrate a custom typesetting engine into the Cocoa text system. On the other hand, an application can subclass `NSATSTypesetter` and override the glyph storage interface to integrate the concrete subclass into its own custom layout system.

`NSTypesetter` methods belong to three categories: glyph storage interface methods, layout phase interface methods, and core typesetter methods. The glyph storage interface methods map to `NSLayoutManager` methods. The typesetter itself calls these methods, and their default implementations call the Cocoa layout manager. An `NSTypesetter` subclass can override these methods to call its own glyph storage facility, in which case it should override all of them. (This does not preclude the overridden method calling its superclass implementation if appropriate.)

The layout phase interface provides control points similar to delegate methods; if implemented, the system invokes these methods to notify an `NSTypesetter` subclass of events in the layout process so it can intervene as needed.

The remainder of the `NSTypesetter` methods are primitive, core typesetter methods. The core typesetter methods correlate with typesetting state attributes; the layout manager calls these methods to store its values before starting the layout process. If you subclass `NSTypesetter` and override the glyph storage interface methods, you can call the core methods to control the typesetter directly.

Glyph Storage Interface

Override these methods to use `NSTypesetter`'s built-in concrete subclass, `NSATSTypesetter`, with a custom glyph storage and layout system other than the Cocoa layout manager and text container mechanism.

`characterRangeForGlyphRange:actualGlyphRange:` (page 3086)
`glyphRangeForCharacterRange:actualCharacterRange:` (page 3091)
`getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:bidiLevels:` (page 3088)
`getLineFragmentRect:usedRect:remainingRect:forStartingGlyphAtIndex:proposedRect:lineSpacing:paragraphSpacingBefore:paragraphSpacingAfter:` (page 3090)
`setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 3102)
`substituteGlyphsInRange:withGlyphs:` (page 3107)
`insertGlyph:atGlyphIndex:characterIndex:` (page 3092)
`deleteGlyphsInRange:` (page 3087)
`setNotShownAttribute:forGlyphRange:` (page 3104)
`setDrawsOutsideLineFragment:forGlyphRange:` (page 3101)
`setLocation:withAdvancements:forStartOfGlyphRange:` (page 3103)
`setAttachmentSize:forGlyphRange:` (page 3099)
`setBidiLevels:forGlyphRange:` (page 3100)

Layout Phase Interface

Override these methods to customize the text layout process, including modifying line fragments, controlling line breaking and hyphenation, and controlling the behavior of tabs and other control glyphs.

`willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 3109)
`shouldBreakLineByWordBeforeCharacterAtIndex:` (page 3106)
`shouldBreakLineByHyphenatingBeforeCharacterAtIndex:` (page 3105)
`hyphenationFactorForGlyphAtIndex:` (page 3091)
`hyphenCharacterForGlyphAtIndex:` (page 3092)
`boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:` (page 3085)

Tasks

Getting a Typesetter

`+ sharedSystemTypesetter` (page 3082)
Returns a shared instance of a reentrant typesetter.

+ sharedSystemTypesetterForBehavior: (page 3082)
Returns a shared instance of a reentrant typesetter that implements typesetting with the specified behavior.

Getting Information About a Typesetter

+ defaultTypesetterBehavior (page 3081)
Returns the default typesetter behavior.

Getting Information About Glyphs

+ printingAdjustmentInLayoutManager:forNominallySpacedGlyphRange:packedGlyphs:count: (page 3081)
Returns the interglyph spacing in the specified range when sent to a printer.

- baselineOffsetInLayoutManager:glyphIndex: (page 3084)
Returns the distance from the bottom of the bounding box of a specified glyph to its baseline.

Managing the Layout Manager

- layoutManager (page 3094)
Returns the layout manager for the text being typeset.

- setUsesFontLeading: (page 3105)
Sets whether the typesetter uses the leading (or line gap) value specified in the font metric information.

- usesFontLeading (page 3108)
Returns whether the typesetter uses the leading (or line gap) value specified in the font metric information of the current font.

- setTypesetterBehavior: (page 3104)
Sets the default typesetter behavior, which affects glyph spacing and line height.

- typesetterBehavior (page 3108)
Returns the current typesetter behavior.

- setHyphenationFactor: (page 3101)
Sets the threshold controlling when hyphenation is attempted.

- hyphenationFactor (page 3091)
Returns the current hyphenation factor.

Managing Text Containers

- currentTextContainer (page 3087)
Returns the text container for the text being typeset.

- textContainers (page 3107)
Returns an array containing the text containers belonging to the current layout manager.

- setLineFragmentPadding: (page 3102)
Sets the amount (in points) by which text is inset within line fragment rectangles.

- `lineFragmentPadding` (page 3095)
  Returns the current line fragment padding, in points.

Mapping Screen and Printer Fonts

- `substituteFontForFont:` (page 3106)
  Returns a screen font suitable for use in place of a given font.

Handling Control Characters

- `textTabForGlyphLocation:writingDirection:maxLocation:` (page 3108)
  Returns the text tab next closest to a given glyph location within the given parameters.
- `actionForControlCharacterAtIndex:` (page 3083)
  Returns the action associated with a control character.

Bidirectional Text Processing

- `setBidiProcessingEnabled:` (page 3100)
  Controls whether the typesetter performs bidirectional text processing.
- `bidiProcessingEnabled` (page 3085)
  Returns whether bidirectional text processing is enabled.

Accessing Paragraph Typesetting Information

- `currentParagraphStyle` (page 3087)
  Returns the paragraph style object for the text being typeset.
- `setAttributedString:` (page 3099)
  Sets the text backing store on which this typesetter operates.
- `attributedString` (page 3083)
  Returns the text backing store, usually an instance of `NSTextStorage`.
- `setParagraphGlyphRange:separatorGlyphRange:` (page 3104)
  Sets the current glyph range being processed.
- `paragraphGlyphRange` (page 3096)
  Returns the glyph range currently being processed.
- `paragraphSeparatorGlyphRange` (page 3097)
  Returns the current paragraph separator range.
- `paragraphCharacterRange` (page 3096)
  Returns the character range currently being processed.
- `paragraphSeparatorCharacterRange` (page 3097)
  Returns the current paragraph separator character range.
- `attributesForExtraLineFragment` (page 3083)
  Returns the attributes used to lay out the extra line fragment.

Paragraph Layout

- `layoutParagraphAtPoint:` (page 3094)
  Lays out glyphs in the current glyph range until the next paragraph separator is reached.
- `beginParagraph` (page 3085)
  Sets up layout parameters at the beginning of a paragraph.
- `endParagraph` (page 3088)
  Sets up layout parameters at the end of a paragraph.
- `beginLineWithGlyphAtIndex:` (page 3084)
  Sets up layout parameters at the beginning of a line during typesetting.
- `endLineWithGlyphRange:` (page 3088)
  Sets up layout parameters at the end of a line during typesetting.

Character Layout

- `layoutCharactersInRange:forLayoutManager:maximumNumberOfLineFragments:` (page 3093)
  Lays out characters in the given character range for the specified layout manager.

Line and Paragraph Spacing

- `lineSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:` (page 3095)
  Returns the line spacing in effect following the specified glyph.
- `paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:` (page 3098)
  Returns the paragraph spacing that is in effect after the specified glyph.
- `paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:` (page 3098)
  Returns the number of points of space—added before a paragraph—that is in effect before the specified glyph.

Glyph Caching

- `setHardInvalidation:forGlyphRange:` (page 3101)
  Sets whether to force the layout manager to invalidate the specified portion of the glyph cache when invalidating layout.

Laying out Glyphs

- `layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:` (page 3094)
  Lays out glyphs in the specified layout manager starting at a specified glyph.
- `boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:` (page 3085)
  Returns the bounding rectangle for the specified control glyph with the specified parameters.

- getLineFragmentRect:usedRect:forParagraphSeparatorGlyphRange:atProposedOrigin: (page 3089)
  Calculates the line fragment rectangle and line fragment used rectangle for blank lines.
- getLineFragmentRect:usedRect:remainingRect:forStartingGlyphAtIndex:proposedRect:lineSpacing:paragraphSpacingBefore:paragraphSpacingAfter: (page 3090)
  Calculates line fragment rectangle, line fragment used rectangle, and remaining rectangle for a line fragment.
- hyphenCharacterForGlyphAtIndex: (page 3092)
  Returns the hyphen character to be inserted after the specified glyph.
- hyphenationFactorForGlyphAtIndex: (page 3091)
  Returns the hyphenation factor in effect at a specified location.
- shouldBreakLineByHyphenatingBeforeCharacterAtIndex: (page 3105)
  Returns whether the line being laid out should be broken by hyphenating at the specified character.
- shouldBreakLineByWordBeforeCharacterAtIndex: (page 3106)
  Returns whether the line being laid out should be broken by a word break at the specified character.
- willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset: (page 3109)
  Called by the typesetter just prior to storing the actual line fragment rectangle location in the layout manager.

Interfacing with Glyph Storage

- characterRangeForGlyphRange:actualGlyphRange: (page 3086)
  Returns the range for the characters in the receiver's text store that are mapped to the specified glyphs.
- deleteGlyphsInRange: (page 3087)
  Deletes the specified glyphs from the glyph cache maintained by the layout manager.
- substituteGlyphsInRange:withGlyphs: (page 3107)
  Replaces the specified glyphs with specified replacement glyphs.
- getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:bidiLevels: (page 3088)
  Extracts the information needed to lay out the provided glyphs from the provided range.
- glyphRangeForCharacterRange:actualCharacterRange: (page 3091)
  Returns the range for the glyphs mapped to the characters of the text store in the specified range.
- setAttachmentSize:forGlyphRange: (page 3099)
  Sets the size the specified glyphs (assumed to be attachments) will be asked to draw themselves at.
- setBidiLevels:forGlyphRange: (page 3100)
  Sets the direction of the specified glyphs for bidirectional text.
- setDrawsOutsideLineFragment:forGlyphRange: (page 3101)
  Sets whether the specified glyphs exceed the bounds of the line fragment in which they are laid out.
- setLineFragmentRect:forGlyphRange:usedRect:baselineOffset: (page 3102)
  Sets the line fragment rectangle where the specified glyphs are laid out.
- setLocation:withAdvancements:forStartOfGlyphRange: (page 3103)
  Sets the location where the specified glyphs are laid out.
- setNotShownAttribute:forGlyphRange: (page 3104)
  Sets whether the specified glyphs are not shown.

`- insertGlyph:atGlyphIndex:characterIndex:` (page 3092) Available in Mac OS X v10.3 through Mac OS X v10.3

Enables the typesetter to insert a new glyph into the stream.

Class Methods

defaultTypesetterBehavior

Returns the default typesetter behavior.

```
+ (NSTypesetterBehavior)defaultTypesetterBehavior
```

Return Value
The default typesetter behavior.

Discussion
Possible return values are described in the `NSTypesetterBehavior` (page 1527) section for `NSLayoutManager`.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSTypesetter.h`

printingAdjustmentInLayoutManager:forNominallySpacedGlyphRange:packedGlyphs:count:

Returns the interglyph spacing in the specified range when sent to a printer.

```
+ (NSSize)printingAdjustmentInLayoutManager:(NSLayoutManager *)layoutMgr
    forNominallySpacedGlyphRange:(NSRange)nominallySpacedGlyphsRange
    packedGlyphs:(const unsigned char *)packedGlyphs
    count:(NSUInteger)packedGlyphsCount
```

Parameters

layoutMgr
The layout manager that will do the drawing.

nominallySpacedGlyphsRange
The range of the glyphs whose spacing is desired.

packedGlyphs
The glyphs as they are packed for sending to be drawn in *layoutMgr*.

packedGlyphsCount
The number of glyphs in *packedGlyphs*.

Return Value
The interglyph spacing in the specified range when sent to a printer. If the font metrics of the font used for displaying text on the screen is different from the font metrics of the font used in printing, then this interglyph spacing may need to be adjusted slightly to match that used on the screen.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTypesetter.h

sharedSystemTypesetter

Returns a shared instance of a reentrant typesetter.

```
+ (id)sharedSystemTypesetter
```

Return Value
The shared system typesetter. This typesetter is reentrant.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTypesetter.h

sharedSystemTypesetterForBehavior:

Returns a shared instance of a reentrant typesetter that implements typesetting with the specified behavior.

```
+ (id)sharedSystemTypesetterForBehavior:(NSTypesetterBehavior)theBehavior
```

Parameters
theBehavior
The desired behavior.

Return Value
A shared instance of a reentrant typesetter that implements typesetting with the specified behavior.

Discussion
Possible return values are described in the `NSTypesetterBehavior` (page 1527) section for `NSLayoutManager`.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `setTypesetterBehavior:` (page 3104)
- `typesetterBehavior` (page 3108)

Declared In
NSTypesetter.h

Instance Methods

actionForControlCharacterAtIndex:

Returns the action associated with a control character.

```
- (NSTypesetterControlCharacterAction)actionForControlCharacterAtIndex:(NSUInteger)charIndex
```

Parameters

charIndex
The index of the control character.

Return Value
The action associated with the control character at *charIndex*.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTypesetter.h

attributedString

Returns the text backing store, usually an instance of NSTextStorage.

```
- (NSAttributedString *)attributedString
```

Return Value
The text backing store.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setAttributedString: (page 3099)

Declared In
NSTypesetter.h

attributesForExtraLineFragment

Returns the attributes used to lay out the extra line fragment.

```
- (NSDictionary *)attributesForExtraLineFragment
```

Return Value
A dictionary of attributes used to lay out the extra line fragment.

Discussion
The default implementation tries to use the NSTextView method typingAttributes (page 2963) if possible; otherwise, it uses the attributes for the last character.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTypesetter.h

baselineOffsetInLayoutManager:glyphIndex:

Returns the distance from the bottom of the bounding box of a specified glyph to its baseline.

```
- (CGFloat)baselineOffsetInLayoutManager:(NSLayoutManager *)layoutMgr
    glyphIndex:(NSUInteger)glyphIndex
```

Parameters

layoutMgr
The layout manager used for the drawing.

glyphIndex
The index of the glyph in question.

Return Value
The distance from the bottom of the bounding box of the glyph in *layoutMgr* specified by *glyphIndex* to its baseline.

Discussion
The text system uses this value to calculate the vertical position of underlines.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTypesetter.h

beginLineWithGlyphAtIndex:

Sets up layout parameters at the beginning of a line during typesetting.

```
- (void)beginLineWithGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

glyphIndex
The index of the first glyph to be laid out in the line.

Discussion
Concrete subclass implementations of `layoutParagraphAtPoint:` (page 3094) should invoke this method at the beginning of each line.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- endLineWithGlyphRange:` (page 3088)

Declared In
NSTypesetter.h

beginParagraph

Sets up layout parameters at the beginning of a paragraph.

```
- (void)beginParagraph
```

Discussion
Concrete subclasses should invoke this method at the beginning of their `layoutParagraphAtPoint:` (page 3094) implementation.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- endParagraph` (page 3088)

Declared In
`NSTypesetter.h`

bidiProcessingEnabled

Returns whether bidirectional text processing is enabled.

```
- (BOOL)bidiProcessingEnabled
```

Return Value
`YES` if bidirectional text processing is enabled, `NO` otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setBidiProcessingEnabled:` (page 3100)

Declared In
`NSTypesetter.h`

boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:

Returns the bounding rectangle for the specified control glyph with the specified parameters.

```
- (NSRect)boundingBoxForControlGlyphAtIndex:(NSUInteger)glyphIndex
    forTextContainer:(NSTextContainer *)textContainer
    proposedLineFragment:(NSRect)proposedRect glyphPosition:(NSPoint)glyphPosition
     characterIndex:(NSUInteger)charIndex
```

Parameters

glyphIndex
The index of the control glyph in question.

textContainer
The text container to use to calculate the position.

proposedRect
The proposed line fragment rectangle.

glyphPosition
The position of the glyph in *textContainer*.

charIndex
The character index in *textContainer*.

Return Value
The bounding rectangle of the control glyph at *glyphIndex*, at the given *glyphPosition* and character index *charIndex*, in *textContainer*.

Discussion
The typesetter calls this method when it encounters a control glyph. The default behavior is to return zero width for control glyphs. A subclass can override this method to do something different, such as implement a way to display control characters.

`NSGlyphGenerator` can choose whether or not to map control characters to `NSControlGlyph`. Tab characters, for example, do not use this facility.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTypesetter.h`

characterRangeForGlyphRange:actualGlyphRange:

Returns the range for the characters in the receiver's text store that are mapped to the specified glyphs.

```
- (NSRange)characterRangeForGlyphRange:(NSRange)glyphRange
    actualGlyphRange:(NSRangePointer)actualGlyphRange
```

Parameters

glyphRange
The range of glyphs.

actualGlyphRange
On return, the range of all glyphs mapped to the characters in the receiver's text store. May be `NULL`.

Return Value
The range for the characters in the receiver's text store that are mapped to the glyphs in *glyphRange*.

Discussion
A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- glyphRangeForCharacterRange:actualCharacterRange:` (page 3091)

Declared In
`NSTypesetter.h`

currentParagraphStyle

Returns the paragraph style object for the text being typeset.

```
- (NSParagraphStyle *)currentParagraphStyle
```

Return Value
The paragraph style object for the text being typeset. This value is valid only while the typesetter is performing layout. More specifically, it's valid only when called inside `layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:` (page 3094).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTypesetter.h`

currentTextContainer

Returns the text container for the text being typeset.

```
- (NSTextContainer *)currentTextContainer
```

Return Value
The text container for the text being typeset. This value is valid only while the typesetter is performing layout. More specifically, it's valid only when called inside `layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:` (page 3094).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- textContainers` (page 3107)

Declared In
`NSTypesetter.h`

deleteGlyphsInRange:

Deletes the specified glyphs from the glyph cache maintained by the layout manager.

```
- (void)deleteGlyphsInRange:(NSRange)glyphRange
```

Parameters
glyphRange
The range of glyphs to be deleted.

Discussion
A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.4 and later.

See Also
`insertGlyph:atGlyphIndex:characterIndex:` (page 3092)

Declared In
`NSTypesetter.h`

endLineWithGlyphRange:

Sets up layout parameters at the end of a line during typesetting.

```
- (void)endLineWithGlyphRange:(NSRange)lineGlyphRange
```

Parameters

lineGlyphRange
The range of glyphs laid out in the line.

Discussion
Concrete subclass implementations of `layoutParagraphAtPoint:` (page 3094) should invoke this method at the end of each line.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- beginLineWithGlyphAtIndex:` (page 3084)

Declared In
`NSTypesetter.h`

endParagraph

Sets up layout parameters at the end of a paragraph.

```
- (void)endParagraph
```

Discussion
Concrete subclasses should invoke this method at the end of their `layoutParagraphAtPoint:` (page 3094) implementation.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- beginParagraph` (page 3085)

Declared In
`NSTypesetter.h`

getGlyphsInRange:glyphs:characterIndexes:glyphInscriptions:elasticBits:bidiLevels:

Extracts the information needed to lay out the provided glyphs from the provided range.

```
- (NSUInteger)getGlyphsInRange:(NSRange)glyphsRange glyphs:(NSGlyph *)glyphBuffer
    characterIndexes:(NSUInteger *)charIndexBuffer
   glyphInscriptions:(NSGlyphInscription *)inscribeBuffer elasticBits:(BOOL
   *)elasticBuffer bidiLevels:(unsigned char *)bidiLevelBuffer
```

Parameters

glyphsRange

The range of glyphs.

glyphBuffer

The glyphs to lay out.

charIndexBuffer

The original characters for the glyphs. Note that a glyph at index 1 is not necessarily mapped to the character at index 1, because a glyph may be for a ligature or accent.

inscribeBuffer

The inscription attributes for each glyph, which are used to layout characters that are combined together.

elasticBuffer

Contains a Boolean value indicating whether a glyph is elastic for each glyph. An elastic glyph can be made longer at the end of a line or when needed for justification.

bidiLevelBuffer

Contains the bidirectional level value generated by `NSGlyphGenerator`, in case a subclass chooses to use this value.

Discussion

A subclass can override this method to interact with custom glyph storage.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSTypesetter.h`

getLineFragmentRect:usedRect:forParagraphSeparatorGlyphRange:atProposedOrigin:

Calculates the line fragment rectangle and line fragment used rectangle for blank lines.

```
- (void)getLineFragmentRect:(NSRectPointer)lineFragmentRect
    usedRect:(NSRectPointer)lineFragmentUsedRect
    forParagraphSeparatorGlyphRange:(NSRange)paragraphSeparatorGlyphRange
    atProposedOrigin:(NSPoint)lineOrigin
```

Parameters

lineFragmentRect

On return, the calculated line fragment rectangle.

lineFragmentUsedRect

On return, the used rectangle (the portion of the line fragment rectangle that actually contains marks).

paragraphSeparatorGlyphRange

The range of glyphs under consideration. A *paragraphSeparatorGlyphRange* with length 0 indicates an extra line fragment (which occurs if the last character in the paragraph is a line separator).

lineOrigin
The origin point of the line fragment rectangle.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTypesetter.h

getLineFragmentRect:usedRect:remainingRect:forStartingGlyphAtIndex:proposedRect:lineSpacing:paragraphSpacingBefore:paragraphSpacingAfter:

Calculates line fragment rectangle, line fragment used rectangle, and remaining rectangle for a line fragment.

```
- (void)getLineFragmentRect:(NSRectPointer)lineFragmentRect
    usedRect:(NSRectPointer)lineFragmentUsedRect
    remainingRect:(NSRectPointer)remainingRect
    forStartingGlyphAtIndex:(NSUInteger)startingGlyphIndex
    proposedRect:(NSRect)proposedRect lineSpacing:(CGFloat)lineSpacing
    paragraphSpacingBefore:(CGFloat)paragraphSpacingBefore
    paragraphSpacingAfter:(CGFloat)paragraphSpacingAfter
```

Parameters

lineFragmentRect
On return, the calculated line fragment rectangle.

lineFragmentUsedRect
On return, the used rectangle (the portion of the line fragment rectangle that actually contains marks).

remainingRect
On return, the remaining rectangle of *proposedRect*.

startingGlyphIndex
The glyph index where the line fragment starts.

proposedRect
The proposed rectangle of the line fragment.

lineSpacing
The line spacing.

paragraphSpacingBefore
The spacing before the paragraph.

paragraphSpacingAfter
The spacing after the paragraph.

Discussion
The height of the line fragment is determined using *lineSpacing*, *paragraphSpacingBefore*, and *paragraphSpacingAfter* as well as *proposedRect*. The width for *lineFragmentUsedRect* is set to the *lineFragmentRect* width. In the standard implementation, paragraph spacing is included in the line fragment rectangle but not the line fragment used rectangle; line spacing is included in both.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTypesetter.h

glyphRangeForCharacterRange:actualCharacterRange:

Returns the range for the glyphs mapped to the characters of the text store in the specified range.

```
- (NSRange)glyphRangeForCharacterRange:(NSRange)charRange
    actualCharacterRange:(NSRangePointer)actualCharRange
```

Parameters

charRange

The range of the characters whose glyph range is desired.

actualCharRange

On return, all characters mapped to those glyphs; may be `NULL`.

Return Value
The range for the glyphs mapped to the characters of the text store in *charRange*.

Discussion
A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `characterRangeForGlyphRange:actualGlyphRange:` (page 3086)

Declared In
`NSTypesetter.h`

hyphenationFactor

Returns the current hyphenation factor.

```
- (float)hyphenationFactor
```

Return Value
The hyphenation factor, a value ranging from 0.0 to 1.0 that controls when hyphenation is attempted. By default, the value is 0.0, meaning hyphenation is off. A factor of 1.0 causes hyphenation to be attempted always.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setHyphenationFactor:` (page 3101)

Declared In
`NSTypesetter.h`

hyphenationFactorForGlyphAtIndex:

Returns the hyphenation factor in effect at a specified location.

```
- (float)hyphenationFactorForGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

glyphIndex

The index of the glyph position to examine.

Return Value

The hyphenation factor in effect at *glyphIndex*. The hyphenation factor is a value ranging from 0.0 to 1.0 that controls when hyphenation is attempted. By default, the value is 0.0, meaning hyphenation is off. A factor of 1.0 causes hyphenation to be attempted always.

Discussion

The typesetter calls this method with a proposed hyphenation point for a line break to find the hyphenation factor in effect at that time. A subclass can override this method to customize the text layout process.

Availability

Available in Mac OS X v10.4 and later.

See Also

- hyphenCharacterForGlyphAtIndex: (page 3092)

Declared In

NSTypesetter.h

hyphenCharacterForGlyphAtIndex:

Returns the hyphen character to be inserted after the specified glyph.

```
- (UTF32Char)hyphenCharacterForGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

glyphIndex

The index of the glyph in question.

Return Value

The hyphen character to be inserted after the glyph at *glyphIndex*.

Discussion

The typesetter calls this method before hyphenating. A subclass can override this method to return a different hyphen glyph.

Availability

Available in Mac OS X v10.4 and later.

See Also

- hyphenationFactorForGlyphAtIndex: (page 3091)

Declared In

NSTypesetter.h

insertGlyph:atGlyphIndex:characterIndex:

Enables the typesetter to insert a new glyph into the stream.

```
- (void)insertGlyph:(NSGlyph)glyph atGlyphIndex:(NSUInteger)glyphIndex
    characterIndex:(NSUInteger)charIndex
```

Parameters

glyph

The glyph to insert into the glyph cache.

glyphIndex

The index at which to insert *glyph*.

charIndex

The index of the character that *glyph* maps to. If the glyph is mapped to several characters, *charIndex* should indicate the first character to which it's mapped.

Discussion

The standard typesetter uses this method for inserting hyphenation glyphs. Because this method keeps the glyph caches synchronized, subclasses should always use this method to insert glyphs instead of calling `layoutManager` (page 3094) directly.

A subclass can override this method to interact with custom glyph storage.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSTypesetter.h`

layoutCharactersInRange:forLayoutManager:maximumNumberOfLineFragments:

Lays out characters in the given character range for the specified layout manager.

```
- (NSRange)layoutCharactersInRange:(NSRange)characterRange
    forLayoutManager:(NSLayoutManager *)layoutManager
    maximumNumberOfLineFragments:(NSUInteger)maxNumLines
```

Parameters

characterRange

The range of the characters to lay out.

layoutManager

The layout manager that does the drawing.

maxNumLines

The maximum number of line fragments to lay out. Specify `NSUIntegerMax` for unlimited number of line fragments.

Return Value

The method returns the actual character range that the receiving `NSTypesetter` processed.

Discussion

The layout process can be interrupted when the number of line fragments exceeds *maxNumLines*.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSTypesetter.h`

layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:

Lays out glyphs in the specified layout manager starting at a specified glyph.

```
- (void)layoutGlyphsInLayoutManager:(NSLayoutManager *)layoutMgr
    startingAtGlyphIndex:(NSUInteger)startGlyphIndex
    maxNumberOfLineFragments:(NSUInteger)maxNumLines nextGlyphIndex:(NSUInteger
    *)nextGlyph
```

Parameters

layoutMgr

The layout manager in which to lay out glyphs.

startGlyphIndex

The index of the starting glyph.

maxNumLines

The maximum number of lines to generate. Fewer lines may be laid out if the glyph storage runs out of glyphs.

nextGlyph

On return, set to the index of the next glyph that needs to be laid out.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTypesetter.h

layoutManager

Returns the layout manager for the text being typeset.

```
- (NSLayoutManager *)layoutManager
```

Return Value
The layout manager for the text being typeset. This value is valid only while the typesetter is performing layout. More specifically, it's valid only when called inside `layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:` (page 3094).

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTypesetter.h

layoutParagraphAtPoint:

Lays out glyphs in the current glyph range until the next paragraph separator is reached.

```
- (NSUInteger)layoutParagraphAtPoint:(NSPointPointer)lineFragmentOrigin
```

Parameters

lineFragmentOrigin

The upper-left corner of line fragment rectangle. On return, *lineFragmentOrigin* contains the next origin.

Return Value

The next glyph index; usually the index right after the paragraph separator, but it can be inside the paragraph range if, for example, the end of the text container is reached before the paragraph separator.

Discussion

Concrete subclasses must implement this method. A concrete implementation must invoke `beginParagraph` (page 3085), `beginLineWithGlyphAtIndex:` (page 3084), `endLineWithGlyphRange:` (page 3088), and `endParagraph` (page 3088).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSTypesetter.h`

lineFragmentPadding

Returns the current line fragment padding, in points.

```
- (CGFloat)lineFragmentPadding
```

Return Value

The current line fragment padding, in points; that is, the portion on each end of the line fragment rectangle left blank.

Discussion

Text is inset within the line fragment rectangle by this amount.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- setLineFragmentPadding:` (page 3102)

Declared In

`NSTypesetter.h`

lineSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:

Returns the line spacing in effect following the specified glyph.

```
- (CGFloat)lineSpacingAfterGlyphAtIndex:(NSUInteger)glyphIndex
    withProposedLineFragmentRect:(NSRect)rect
```

Parameters

glyphIndex

The index of the glyph in question.

rect
The proposed line fragment rectangle.

Return Value
the line spacing in effect following the glyph at *glyphIndex*.

Discussion
The `NSATSTypesetter` calls this method to determine the number of points of space to include below the descenders in the used rectangle for the proposed line fragment rectangle *rect*.

Line spacing, also called leading, is an attribute of `NSParagraphStyle`, which you can set on an `NSMutableParagraphStyle` object. A font typically includes a default minimum line spacing metric used if none is set in the paragraph style.

If the typesetter behavior specified in the layout manager is `NSTypesetterOriginalBehavior`, the text system uses the original, private typesetter `NSSimpleHorizontalTypesetter`, which adds the line spacing above the ascender. Similarly, `NSATSTypesetter` adds the line spacing above the ascender if the value is negative.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTypesetter.h`

paragraphCharacterRange

Returns the character range currently being processed.

```
- (NSRange)paragraphCharacterRange
```

Return Value
The character range currently being processed.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `paragraphSeparatorCharacterRange` (page 3097)
- `paragraphSeparatorGlyphRange` (page 3097)
- `paragraphGlyphRange` (page 3096)

Declared In
`NSTypesetter.h`

paragraphGlyphRange

Returns the glyph range currently being processed.

```
- (NSRange)paragraphGlyphRange
```

Return Value
The glyph range currently being processed.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setParagraphGlyphRange:separatorGlyphRange:` (page 3104)
- `paragraphSeparatorGlyphRange` (page 3097)
- `paragraphCharacterRange` (page 3096)
- `paragraphSeparatorCharacterRange` (page 3097)

Declared In
`NSTypesetter.h`

paragraphSeparatorCharacterRange

Returns the current paragraph separator character range.

```
- (NSRange)paragraphSeparatorCharacterRange
```

Return Value
The current paragraph separator character range, which is the full range that contains the current character range and that extends from one paragraph separator character to the next.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `paragraphGlyphRange` (page 3096)
- `paragraphSeparatorGlyphRange` (page 3097)
- `paragraphCharacterRange` (page 3096)

Declared In
`NSTypesetter.h`

paragraphSeparatorGlyphRange

Returns the current paragraph separator range.

```
- (NSRange)paragraphSeparatorGlyphRange
```

Return Value
The current paragraph separator range, which is the full range that contains the current glyph range and that extends from one paragraph separator character to the next.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setParagraphGlyphRange:separatorGlyphRange:` (page 3104)
- `paragraphGlyphRange` (page 3096)
- `paragraphSeparatorCharacterRange` (page 3097)
- `paragraphCharacterRange` (page 3096)

Declared In
NSTypesetter.h

paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:

Returns the paragraph spacing that is in effect after the specified glyph.

```
- (CGFloat)paragraphSpacingAfterGlyphAtIndex:(NSUInteger)glyphIndex
    withProposedLineFragmentRect:(NSRect)rect
```

Parameters

glyphIndex
The index of the glyph in question.

rect
The line fragment rectangle of the last line in the paragraph.

Return Value
The paragraph spacing—that is, the number of points of space added following a paragraph—that is in effect after the glyph specified by *glyphIndex*.

Discussion
The typesetter adds the number of points specified in the return value to the bottom of the line fragment rectangle specified by *rect* (but not to the used line fragment rectangle for that line). Paragraph spacing added after a paragraph correlates to the value returned by the `paragraphSpacing` (page 1875) method of `NSParagraphStyle`, which you can set using the `setParagraphSpacing:` (page 1723) method of `NSMutableParagraphStyle`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:` (page 3098)

Declared In
NSTypesetter.h

paragraphSpacingBeforeGlyphAtIndex:withProposedLineFragmentRect:

Returns the number of points of space—added before a paragraph—that is in effect before the specified glyph.

```
- (CGFloat)paragraphSpacingBeforeGlyphAtIndex:(NSUInteger)glyphIndex
    withProposedLineFragmentRect:(NSRect)rect
```

Parameters

glyphIndex
The index of the glyph in question.

rect
The line fragment rectangle of the first line in the paragraph.

Return Value
The number of points of space—added before a paragraph—that is in effect before the glyph specified by *glyphIndex*.

Discussion
The typesetter adds the number of points specified in the return value to the top of the line fragment rectangle specified by *rect* (but not to the used line fragment rectangle for that line). Paragraph spacing added before a paragraph correlates to the value returned by the `paragraphSpacingBefore` (page 1876) method of `NSParagraphStyle`, which you can set using the `setParagraphSpacingBefore:` (page 1724) method of `NSMutableParagraphStyle`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- paragraphSpacingAfterGlyphAtIndex:withProposedLineFragmentRect:` (page 3098)

Declared In
`NSTypesetter.h`

setAttachmentSize:forGlyphRange:

Sets the size the specified glyphs (assumed to be attachments) will be asked to draw themselves at.

```
- (void)setAttachmentSize:(NSSize)attachmentSize forGlyphRange:(NSRange)glyphRange
```

Parameters

attachmentSize
The size the glyphs in *glyphRange* (assumed to be attachments) will be asked to draw themselves at.

glyphRange
The range of glyphs the attachment size applies to.

Discussion
A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTypesetter.h`

setAttributedString:

Sets the text backing store on which this typesetter operates.

```
- (void)setAttributedString:(NSAttributedString *)attrString
```

Parameters

attrString
The text backing store on which the typesetter should operate.

Special Considerations

Typesetters do not retain the text backing store on which they are operating.

Availability
Available in Mac OS X v10.4 and later.

See Also
- attributedString (page 3083)

Declared In
NSTypesetter.h

setBidiLevels:forGlyphRange:

Sets the direction of the specified glyphs for bidirectional text.

```
- (void)setBidiLevels:(const uint8_t *)levels forGlyphRange:(NSRange)glyphRange
```

Parameters

levels

Values in *levels* can range from 0 to 61 as defined by Unicode Standard Annex #9.

glyphRange

The range of glyphs for which the bidirectional text levels are desired.

Discussion
A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTypesetter.h

setBidiProcessingEnabled:

Controls whether the typesetter performs bidirectional text processing.

```
- (void)setBidiProcessingEnabled:(BOOL)flag
```

Parameters

flag

YES to enable bidirectional text processing, NO to disable it.

Discussion
You can use this method to disable the bidirectional layout stage if you know the paragraph does not need this stage; that is, if the characters in the backing store are in display order.

Availability
Available in Mac OS X v10.4 and later.

See Also
- bidiProcessingEnabled (page 3085)

Declared In
NSTypesetter.h

setDrawsOutsideLineFragment:forGlyphRange:

Sets whether the specified glyphs exceed the bounds of the line fragment in which they are laid out.

```
- (void)setDrawsOutsideLineFragment:(BOOL)flag forGlyphRange:(NSRange)glyphRange
```

Parameters

flag

YES if the glyphs in *glyphRange* exceed the bounds of the line fragment in which they are laid out, NO otherwise.

glyphRange

The range of the glyphs in question.

Discussion

This can happen when text is set at a fixed line height. For example, if the user specifies a fixed line height of 12 points and sets the font size to 24 points, the glyphs will exceed their layout rectangles.

A subclass can override this method to interact with custom glyph storage.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSTypesetter.h

setHardInvalidation:forGlyphRange:

Sets whether to force the layout manager to invalidate the specified portion of the glyph cache when invalidating layout.

```
- (void)setHardInvalidation:(BOOL)flag forGlyphRange:(NSRange)glyphRange
```

Parameters

flag

YES if the layout manager should invalidate the specified portion of the glyph cache, NO otherwise.

glyphRange

The range of glyphs in the cache to be marked for hard invalidation.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSTypesetter.h

setHyphenationFactor:

Sets the threshold controlling when hyphenation is attempted.

```
- (void)setHyphenationFactor:(float)factor
```

Parameters

factor

A frequency factor in the range of 0.0 to 1.0. By default, the value is 0.0, meaning hyphenation is off. A *factor* of 1.0 causes hyphenation to be attempted always.

Availability
Available in Mac OS X v10.4 and later.

See Also
- hyphenationFactor (page 3091)

Declared In
NSTypesetter.h

setLineFragmentPadding:

Sets the amount (in points) by which text is inset within line fragment rectangles.

```
- (void)setLineFragmentPadding:(CGFloat)padding
```

Parameters

padding

The amount (in points) by which text is inset within line fragment rectangles.

Special Considerations

Line fragment padding isn't a suitable means for expressing margins; you should set the text view's position and size for document margins or the paragraph margin attributes for text margins.

Availability
Available in Mac OS X v10.4 and later.

See Also
- lineFragmentPadding (page 3095)

Declared In
NSTypesetter.h

setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:

Sets the line fragment rectangle where the specified glyphs are laid out.

```
- (void)setLineFragmentRect:(NSRect)fragmentRect forGlyphRange:(NSRange)glyphRange
    usedRect:(NSRect)usedRect baselineOffset:(CGFloat)baselineOffset
```

Parameters

fragmentRect

The line fragment rectangle where the glyphs in *glyphRange* are laid out.

glyphRange

The range of the specified glyphs.

usedRect
The portion of *fragmentRect*, in the NSTextContainer object's coordinate system, that actually contains glyphs or other marks that are drawn (including the text container's line fragment padding). The *usedRect* must be equal to or contained within *fragmentRect*.

baselineOffset
The vertical distance in pixels from the line fragment origin to the baseline on which the glyphs align.

Discussion
The exact positions of the glyphs must be set after the line fragment rectangle with `setLocation:forStartOfGlyphRange:`.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTypesetter.h`

setLocation:withAdvancements:forStartOfGlyphRange:

Sets the location where the specified glyphs are laid out.

```
- (void)setLocation:(NSPoint)location withAdvancements:(const CGFloat *)advancements
    forStartOfGlyphRange:(NSRange)glyphRange
```

Parameters
location
The location where the glyphs in *glyphRange* are laid out. The x-coordinate of *location* is expressed relative to the line fragment rectangle origin, and the y-coordinate is expressed relative to the baseline previously specified by `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 3102).

advancements
The nominal glyph advance width specified in the font metric information.

glyphRange
The range of glyphs whose layout location is being set. This series of glyphs can be displayed with a single PostScript `show` operation (a nominal range).

Discussion
Setting the location for a series of glyphs implies that the glyphs preceding it can't be included in a single `show` operation.

Before setting the location for a glyph range, you must specify line fragment rectangle with `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:`.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTypesetter.h`

setNotShownAttribute:forGlyphRange:

Sets whether the specified glyphs are not shown.

```
- (void)setNotShownAttribute:(BOOL)flag forGlyphRange:(NSRange)glyphRange
```

Parameters

flag

YES if the glyphs in *glyphRange* are not shown, NO if they are shown.

glyphRange

The range of glyphs in question.

Discussion

For example, a tab or newline character doesn't leave any marks; it just indicates where following glyphs are laid out.

A subclass can override this method to interact with custom glyph storage.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSTypesetter.h

setParagraphGlyphRange:separatorGlyphRange:

Sets the current glyph range being processed.

```
- (void)setParagraphGlyphRange:(NSRange)paragraphRange
    separatorGlyphRange:(NSRange)paragraphSeparatorRange
```

Parameters

paragraphRange

The current glyph range being processed.

paragraphSeparatorRange

The range of the paragraph separator character or characters.

Availability

Available in Mac OS X v10.4 and later.

See Also

- paragraphGlyphRange (page 3096)
- paragraphSeparatorGlyphRange (page 3097)

Declared In

NSTypesetter.h

setTypesetterBehavior:

Sets the default typesetter behavior, which affects glyph spacing and line height.

```
- (void)setTypesetterBehavior:(NSTypesetterBehavior)behavior
```

Parameters

behavior

The new behavior.

Availability
Available in Mac OS X v10.4 and later.

See Also
- typesetterBehavior (page 3108)

Declared In
NSTypesetter.h

setUsesFontLeading:

Sets whether the typesetter uses the leading (or line gap) value specified in the font metric information.

```
- (void)setUsesFontLeading:(BOOL)flag
```

Parameters

flag

YES to use the information in the font metrics, NO to ignore it.

Availability
Available in Mac OS X v10.4 and later.

See Also
- usesFontLeading (page 3108)

Declared In
NSTypesetter.h

shouldBreakLineByHyphenatingBeforeCharacterAtIndex:

Returns whether the line being laid out should be broken by hyphenating at the specified character.

```
- (BOOL)shouldBreakLineByHyphenatingBeforeCharacterAtIndex:(NSUInteger)charIndex
```

Parameters

charIndex

The index of the character just after the proposed hyphenation would occur.

Return Value
YES if the line should be broken by hyphenating, NO otherwise.

Discussion
The typesetter calls this method, if implemented by a subclass, before breaking a line by hyphenating before the character at *charIndex*, enabling the subclass to control line breaking.

A subclass can override this method to customize the text layout process. If the method returns NO, the typesetter continues looking for a break point.

Availability
Available in Mac OS X v10.4 and later.

See Also
- shouldBreakLineByWordBeforeCharacterAtIndex: (page 3106)

Declared In
NSTypesetter.h

shouldBreakLineByWordBeforeCharacterAtIndex:

Returns whether the line being laid out should be broken by a word break at the specified character.

```
- (BOOL)shouldBreakLineByWordBeforeCharacterAtIndex:(NSUInteger)charIndex
```

Parameters
charIndex
The index of the character just after the proposed word break would occur.

Return Value
YES if the line should be broken by a word break, NO otherwise.

Discussion
The typesetter calls this method, if implemented by a subclass, before breaking a line by word wrapping before the character at *charIndex*, enabling the subclass to control line breaking.

A subclass can override this method to customize the text layout process. If the method returns NO, the typesetter continues looking for a break point.

Availability
Available in Mac OS X v10.4 and later.

See Also
- shouldBreakLineByHyphenatingBeforeCharacterAtIndex: (page 3105)

Declared In
NSTypesetter.h

substituteFontForFont:

Returns a screen font suitable for use in place of a given font.

```
- (NSFont *)substituteFontForFont:(NSFont *)originalFont
```

Parameters
originalFont
The original font.

Return Value
A screen font suitable for use in place of *originalFont*. This method returns *originalFont* if a screen font can't be used or isn't available.

Discussion
A screen font can only be substituted if the receiver is set to use screen fonts and if no text view associated with the receiver is scaled or rotated.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTypesetter.h

substituteGlyphsInRange:withGlyphs:

Replaces the specified glyphs with specified replacement glyphs.

```
- (void)substituteGlyphsInRange:(NSRange)glyphRange withGlyphs:(NSGlyph *)glyphs
```

Parameters

glyphRange
The range of glyphs to be substituted.

glyphs
The glyphs to substitute for the glyphs in `glyphRange`.

Discussion
This method does not alter the glyph-to-character mapping or invalidate layout information.

A subclass can override this method to interact with custom glyph storage.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSTypesetter.h

textContainers

Returns an array containing the text containers belonging to the current layout manager.

```
- (NSArray *)textContainers
```

Return Value
An array containing the text containers belonging to the current layout manager. This value is valid only while the typesetter is performing layout. More specifically, it's valid only when called inside `layoutGlyphsInLayoutManager:startingAtGlyphIndex:maxNumberOfLineFragments:nextGlyphIndex:` (page 3094).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `layoutManager` (page 3094)
- `currentTextContainer` (page 3087)

Declared In
NSTypesetter.h

textTabForGlyphLocation:writingDirection:maxLocation:

Returns the text tab next closest to a given glyph location within the given parameters.

```
- (NSTextTab *)textTabForGlyphLocation:(CGFloat)glyphLocation
    writingDirection:(NSWritingDirection)direction maxLocation:(CGFloat)maxLocation
```

Parameters

glyphLocation

The location at which to start searching.

direction

The direction in which to search.

maxLocation

The maximum location for the search.

Return Value

The text tab next closest to *glyphLocation*, indexing in *direction* but not beyond *maxLocation*.

Discussion

The typesetter calls this method whenever it finds a tab character. To determine the width to advance the next glyph, the typesetter examines the `NSParagraphStyle` object's tab array and the default tab interval.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSTypesetter.h`

typesetterBehavior

Returns the current typesetter behavior.

```
- (NSTypesetterBehavior)typesetterBehavior
```

Return Value

The current typesetter behavior.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- setTypesetterBehavior:` (page 3104)

Declared In

`NSTypesetter.h`

usesFontLeading

Returns whether the typesetter uses the leading (or line gap) value specified in the font metric information of the current font.

```
- (BOOL)usesFontLeading
```

Return Value
`YES` if it uses the information in the font metrics, `NO` otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setUsesFontLeading:` (page 3105)

Declared In
`NSTypesetter.h`

willSetLineFragmentRect:forGlyphRange:usedRect:baselineOffset:

Called by the typesetter just prior to storing the actual line fragment rectangle location in the layout manager.

```
- (void)willSetLineFragmentRect:(NSRectPointer)lineRect
    forGlyphRange:(NSRange)glyphRange usedRect:(NSRectPointer)usedRect
    baselineOffset:(CGFloat *)baselineOffset
```

Parameters

lineRect
The rectangle in which the glyphs in *glyphRange* are laid out.

glyphRange
The range of the glyphs to lay out.

usedRect
The portion of *lineRect*, in the NSTextContainer object's coordinate system, that actually contains glyphs or other marks that are drawn (including the text container's line fragment padding). The *usedRect* must be equal to or contained within *lineRect*.

baselineOffset
The vertical distance in pixels from the line fragment origin to the baseline on which the glyphs align.

Discussion
Called by the typesetter just prior to calling `setLineFragmentRect:forGlyphRange:usedRect:baselineOffset:` (page 3102) which stores the actual line fragment rectangle location in the layout manager.

A subclass can override this method to customize the text layout process. For example, it could change the shape of the line fragment rectangle. The subclass is responsible for ensuring that the modified rectangle remains valid (for example, that it lies within the text container).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSTypesetter.h`

Constants

NSTypesetterControlCharacterAction

The following constants are possible values returned by the `actionForControlCharacterAtIndex:` (page 3083) method to determine the action associated with a control character.

```
enum {
    NSTypesetterZeroAdvancementAction = (1 << 0),
    NSTypesetterWhitespaceAction = (1 << 1),
    NSTypesetterHorizontalTabAction = (1 << 2),
    NSTypesetterLineBreakAction = (1 << 3),
    NSTypesetterParagraphBreakAction = (1 << 4),
    NSTypesetterContainerBreakAction = (1 << 5)
};
typedef NSUInteger NSTypesetterControlCharacterAction;
```

Constants

`NSTypesetterZeroAdvancementAction`

Glyphs with this action are filtered out from layout `(notShownAttribute == YES)`.

Available in Mac OS X v10.4 and later.

Declared in `NSTypesetter.h`.

`NSTypesetterWhitespaceAction`

The width for glyphs with this action are determined by `boundingBoxForControlGlyphAtIndex:forTextContainer:proposedLineFragment:glyphPosition:characterIndex:` (page 3085), if the method is implemented; otherwise, same as `NSTypesetterZeroAdvancementAction`.

Available in Mac OS X v10.4 and later.

Declared in `NSTypesetter.h`.

`NSTypesetterHorizontalTabAction`

Treated as tab character.

Available in Mac OS X v10.4 and later.

Declared in `NSTypesetter.h`.

`NSTypesetterLineBreakAction`

Causes line break.

Available in Mac OS X v10.4 and later.

Declared in `NSTypesetter.h`.

`NSTypesetterParagraphBreakAction`

Causes paragraph break; the value returned by `firstLineHeadIndent` (page 1872) is the advancement used for the following glyph.

Available in Mac OS X v10.4 and later.

Declared in `NSTypesetter.h`.

`NSTypesetterContainerBreakAction`

Causes container break.

Available in Mac OS X v10.4 and later.

Declared in `NSTypesetter.h`.

NSURL Additions Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSPasteboard.h

Overview

The Application Kit extends Foundation's NSURL class by adding support for pasteboards. NSWorkspace provides `openURL:` (page 3473) to open a location specified by a URL.

Tasks

Working with Pasteboards

`+ URLFromPasteboard:` (page 3111)
Reads an NSURL object off of *pasteboard*. Returns `nil` if *pasteboard* does not contain data of type `NSURLPboardType`.

`- writeToPasteboard:` (page 3112)
Writes the receiver to *pasteboard*. You must declare an `NSURLPboardType` data type for *pasteboard* before invoking this method; otherwise it returns without doing anything.

Class Methods

URLFromPasteboard:

Reads an NSURL object off of *pasteboard*. Returns `nil` if *pasteboard* does not contain data of type `NSURLPboardType`.

```
+ (NSURL *)URLFromPasteboard:(NSPasteboard *)pasteboard
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- writeToPasteboard: (page 3112)

Related Sample Code
CocoaDragAndDrop
NineSlice
QTMetadataEditor
SourceView
With and Without Bindings

Declared In
NSPasteboard.h

Instance Methods

writeToPasteboard:

Writes the receiver to *pasteboard*. You must declare an NSURLPboardType data type for *pasteboard* before invoking this method; otherwise it returns without doing anything.

```
- (void)writeToPasteboard:(NSPasteboard *)pasteboard
```

Availability
Available in Mac OS X v10.0 and later.

See Also
+ URLFromPasteboard: (page 3111)
- declareTypes:owner: (page 1893) (NSPasteboard)

Related Sample Code
With and Without Bindings

Declared In
NSPasteboard.h

NSUserDefaultsController Class Reference

Inherits from	NSController : NSObject
Conforms to	NSCoding (NSController) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSUserDefaultsController.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Cocoa Bindings Programming Topics
Related sample code	DemoMonkey Dicey ImageApp Sketch+Accessibility TemperatureConverter

Overview

NSUserDefaultsController is a Cocoa bindings compatible controller class. Properties of the shared instance of this class can be bound to user interface elements to access and modify values stored in NSUserDefaults.

Tasks

Obtaining the Shared Instance

`+ sharedUserDefaultsController` (page 3114)
Returns the shared instance of NSUserDefaultsController, creating it if necessary.

Initializing a User Defaults Controller

`- initWithDefaults:initialValues:` (page 3116)
Returns an initialized NSUserDefaultsController object using the NSUserDefaults instance specified in *defaults* and the initial default values contained in the *initialValues* dictionary.

Managing User Defaults Values

- `defaults` (page 3115)
  Returns the instance of NSUserDefaults in use by the receiver.
- `setInitialValues:` (page 3118)
  Sets the receiver's initial values to *initialValues*.
- `hasUnappliedChanges` (page 3115)
  Returns whether the receiver has user default values that have not been saved to NSUserDefaults.
- `initialValues` (page 3116)
  Returns a dictionary containing the receiver's initial default values.
- `setAppliesImmediately:` (page 3118)
  Sets whether any changes made to the receiver's user default properties are saved immediately.
- `appliesImmediately` (page 3115)
  Returns whether any changes made to bound user default properties are saved immediately.
- `values` (page 3119)
  Returns a key value coding compliant object that is used to access the user default properties.
- `revert:` (page 3117)
  Causes the receiver to discard any unsaved changes to bound user default properties, restoring their previous values.
- `revertToInitialValues:` (page 3117)
  Causes the receiver to discard all edits and replace the values of all the user default properties with any corresponding values in the `initialValues` (page 3116) dictionary.
- `save:` (page 3117)
  Saves the values of the receiver's user default properties.

Class Methods

sharedUserDefaultsController

Returns the shared instance of NSUserDefaultsController, creating it if necessary.

```
+ (id)sharedUserDefaultsController
```

Discussion
This instance has no initial values, and uses `[NSUserDefaults standardUserDefaults]` to create the defaults. An application can get this object when an application launches and configure it as required.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
DemoMonkey
Dicey
ImageApp
Sketch+Accessibility

Declared In
NSUserDefaultsController.h

Instance Methods

appliesImmediately

Returns whether any changes made to bound user default properties are saved immediately.

```
- (BOOL)appliesImmediately
```

Discussion
Default is YES.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setAppliesImmediately: (page 3118)

Declared In
NSUserDefaultsController.h

defaults

Returns the instance of NSUserDefaults in use by the receiver.

```
- (NSUserDefaults *)defaults
```

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
ImageApp

Declared In
NSUserDefaultsController.h

hasUnappliedChanges

Returns whether the receiver has user default values that have not been saved to NSUserDefaults.

```
- (BOOL)hasUnappliedChanges
```

Discussion
This property is observable using key-value observing.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `appliesImmediately` (page 3115)
- `setAppliesImmediately:` (page 3118)

Declared In
`NSUserDefaultsController.h`

initialValues

Returns a dictionary containing the receiver's initial default values.

```
- (NSDictionary *)initialValues
```

Discussion
These values are used when is no value found for the bound property in `defaults` (page 3115).

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setInitialValues:` (page 3118)
- `revertToInitialValues:` (page 3117)

Declared In
`NSUserDefaultsController.h`

initWithDefaults:initialValues:

Returns an initialized NSUserDefaultsController object using the NSUserDefaults instance specified in *defaults* and the initial default values contained in the *initialValues* dictionary.

```
- (id)initWithDefaults:(NSUserDefaults *)defaults initialValues:(NSDictionary
    *)initialValues
```

Discussion
If *defaults* is `nil`, the receiver uses `[NSUserDefaults standardUserDefaults]`.

This method is the designated initializer.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSUserDefaultsController.h`

revert:

Causes the receiver to discard any unsaved changes to bound user default properties, restoring their previous values.

```
- (void)revert:(id)sender
```

Discussion
The receiver invokes `discardEditing` (page 3679) on any currently registered editors. The *sender* is typically the object that invoked this method.

If `appliesImmediately` (page 3115) is `YES`, this method only causes any bound editors with uncommitted changes to discard their edits.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `revertToInitialValues:` (page 3117)

Declared In
`NSUserDefaultsController.h`

revertToInitialValues:

Causes the receiver to discard all edits and replace the values of all the user default properties with any corresponding values in the `initialValues` (page 3116) dictionary.

```
- (void)revertToInitialValues:(id)sender
```

Discussion
This effectively sets the preferences that a user can change to their "out-of-the-box" values. This method has no effect if initial values were not specified. The *sender* is typically the object that invoked this method.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `initialValues` (page 3116)
- `revert:` (page 3117)

Declared In
`NSUserDefaultsController.h`

save:

Saves the values of the receiver's user default properties.

```
- (void)save:(id)sender
```

Discussion
This method has no effect if `appliesImmediately` (page 3115) returns `YES`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSUserDefaultsController.h

setAppliesImmediately:

Sets whether any changes made to the receiver's user default properties are saved immediately.

```
- (void)setAppliesImmediately:(BOOL)flag
```

Discussion
The default is YES.

Availability
Available in Mac OS X v10.3 and later.

See Also
- appliesImmediately (page 3115)

Declared In
NSUserDefaultsController.h

setInitialValues:

Sets the receiver's initial values to *initialValues*.

```
- (void)setInitialValues:(NSDictionary *)initialValues
```

Discussion
These values are used when a user default properties has no value in NSUserDefaults and by revertToInitialValues: (page 3117).

The initial values must be set before loading a nib that uses the receiver, as those values may be referenced at load time. It is good practice to set the initial values–along with registering any defaults for the applications–in the initialize class method of your preference dialog controller, or the application delegate.

Availability
Available in Mac OS X v10.3 and later.

See Also
- defaults (page 3115)
- initialValues (page 3116)

Related Sample Code
DemoMonkey
Sketch+Accessibility

Declared In
NSUserDefaultsController.h

values

Returns a key value coding compliant object that is used to access the user default properties.

```
- (id)values
```

Discussion
If present the value for the property in `defaults` (page 3115) is returned, otherwise a corresponding value in `initialValues` (page 3116) is returned.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSUserDefaultsController.h`

NSView Class Reference

Inherits from	NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSView.h AppKit/NSClipView.h
Companion guides	View Programming Guide Cocoa Drawing Guide Cocoa Event-Handling Guide Drag and Drop Programming Topics for Cocoa Printing Programming Topics for Cocoa
Related sample code	MatrixMixerTest Quartz Composer WWDC 2005 TextEdit QuickLookSketch Sketch+Accessibility Sketch-112

Class at a Glance

`NSView` is a class that defines the basic drawing, event-handling, and printing architecture of an application. You typically don't interact with the `NSView` API directly; rather, your custom view classes inherit from `NSView` and override many of its methods, which are invoked automatically by the Application Kit. If you're not creating a custom view class, there are few methods you need to use.

Principal Attributes

- Event handling
- Integrated display to screen and printer
- Flexible coordinate systems
- Icon dragging

Commonly Used Methods

`frame` (page 3175)
Returns the location and size of the `NSView` object.

`bounds` (page 3150)
Returns the internal origin and size of the `NSView` object.

`setNeedsDisplay:` (page 3225)
Marks the `NSView` object as needing to be redrawn

`window` (page 3248)
Returns the `NSWindow` object that contains the `NSView` object.

`drawRect:` (page 3170)
Draws the `NSView` object. (All subclasses must implement this method, but it's rarely invoked explicitly.)

Overview

The `NSView` class that provides a structure for drawing, printing, and handling events.

`NSView` objects (also know, simply, as view objects or views) are arranged within an `NSWindow` object, in a nested hierarchy of subviews. A view object claims a rectangular region of its enclosing superview, is responsible for all drawing within that region, and is eligible to receive mouse events occurring in it as well. In addition to these major responsibilities, `NSView` handles dragging of icons and works with the `NSScrollView` class to support efficient scrolling.

Most of the functionality of `NSView` either is automatically invoked by the Application Kit, or is available in Interface Builder. Unless you're implementing a concrete subclass of `NSView` or working intimately with the content of the view hierarchy at runtime, you don't need to know much about this class's interface. See "Commonly Used Methods" (page 3122) for methods you might use regardless.

For more information on how `NSView` instances handle event and action messages, see *Cocoa Event-Handling Guide*. For more information on displaying tooltips and contextual menus, see "Displaying Contextual Menus" (page 1605) and "Managing Tooltips" (page 3286).

Subclassing Notes

`NSView` is perhaps the most important class in the Application Kit when it comes to subclassing and inheritance. Most user-interface objects you see in a Cocoa application are objects that inherit from `NSView`. If you want to create an object that draws itself in a special way, or that responds to mouse clicks in a special way, you would create a custom subclass of `NSView` (or of a class that inherits from `NSView`). Subclassing `NSView` is such a common and important procedure that several technical documents describe how to both draw in custom subclasses and respond to events in custom subclasses. See *Cocoa Drawing Guide* and *Cocoa Event-Handling Guide* (especially ""Handling Mouse Events"" and ""Mouse Events" in *Cocoa Event-Handling Guide*").

Tasks

Creating Instances

- `initWithFrame:` (page 3179)
  Initializes and returns a newly allocated `NSView` object with a specified frame rectangle.

Managing the View Hierarchy

- `superview` (page 3235)
  Returns the receiver's superview, or `nil` if it has none.
- `setSubviews:` (page 3228)
  Sets the receiver's subviews to the specified subviews.
- `subviews` (page 3235)
  Return the receiver's immediate subviews.
- `window` (page 3248)
  Returns the receiver's window object, or `nil` if it has none.
- `addSubview:` (page 3139)
  Adds a view to the receiver's subviews so it's displayed above its siblings.
- `addSubview:positioned:relativeTo:` (page 3140)
  Inserts a view among the receiver's subviews so it's displayed immediately above or below another view.
- `didAddSubview:` (page 3163)
  Overridden by subclasses to perform additional actions when subviews are added to the receiver.
- `removeFromSuperview` (page 3202)
  Unlinks the receiver from its superview and its window, removes it from the responder chain, and invalidates its cursor rectangles.
- `removeFromSuperviewWithoutNeedingDisplay` (page 3203)
  Unlinks the receiver from its superview and its window and removes it from the responder chain, but does not invalidate its cursor rectangles to cause redrawing.
- `replaceSubview:with:` (page 3205)
  Replaces one of the receiver's subviews with another view.
- `isDescendantOf:` (page 3181)
  Returns `YES` if the receiver is a subview of a given view or if it's identical to that view; otherwise, it returns `NO`.
- `opaqueAncestor` (page 3193)
  Returns the receiver's closest opaque ancestor (including the receiver itself).
- `ancestorSharedWithView:` (page 3147)
  Returns the closest ancestor shared by the receiver and a given view.
- `sortSubviewsUsingFunction:context:` (page 3234)
  Orders the receiver's immediate subviews using the specified comparator function.
- `viewDidMoveToSuperview` (page 3241)
  Informs the receiver that its superview has changed (possibly to `nil`).

- `viewDidMoveToWindow` (page 3241)
  Informs the receiver that it has been added to a new view hierarchy.
- `viewWillMoveToSuperview:` (page 3243)
  Informs the receiver that its superview is about to change to the specified superview (which may be `nil`).
- `viewWillMoveToWindow:` (page 3243)
  Informs the receiver that it's being added to the view hierarchy of the specified window object (which may be `nil`).
- `willRemoveSubview:` (page 3247)
  Overridden by subclasses to perform additional actions before subviews are removed from the receiver.
- `enclosingMenuItem` (page 3172)
  Returns the menu item containing the receiver or any of its superviews in the view hierarchy.

Searching by Tag

- `viewWithTag:` (page 3245)
  Returns the receiver's nearest descendant (including itself) with a specific tag, or `nil` if no subview has that tag.
- `tag` (page 3236)
  Returns the receiver's tag, an integer that you can use to identify view objects in your application.

Modifying the Frame Rectangle

- `setFrame:` (page 3218)
  Sets the receiver's frame rectangle to the specified rectangle.
- `frame` (page 3175)
  Returns the receiver's frame rectangle, which defines its position in its superview.
- `setFrameOrigin:` (page 3220)
  Sets the origin of the receiver's frame rectangle to the specified point, effectively repositioning it within its superview.
- `setFrameSize:` (page 3221)
  Sets the size of the receiver's frame rectangle to the specified dimensions, resizing it within its superview without affecting its coordinate system.
- `setFrameRotation:` (page 3221)
  Sets the rotation of the receiver's frame rectangle to a specified degree value, rotating it within its superview without affecting its coordinate system.
- `frameRotation` (page 3176)
  Returns the angle, in degrees, of the receiver's frame relative to its superview's coordinate system.

Modifying the Bounds Rectangle

- `setBounds:` (page 3214)
  Sets the receiver's bounds rectangle.

- `bounds` (page 3150)
  Returns the receiver's bounds rectangle, which expresses its location and size in its own coordinate system.
- `setBoundsOrigin:` (page 3214)
  Sets the origin of the receiver's bounds rectangle to a specified point,
- `setBoundsSize:` (page 3216)
  Sets the size of the receiver's bounds rectangle to specified dimensions, inversely scaling its coordinate system relative to its frame rectangle.
- `setBoundsRotation:` (page 3215)
  Sets the rotation of the receiver's bounds rectangle to a specific degree value.
- `boundsRotation` (page 3151)
  Returns the angle, in degrees, of the receiver's bounds rectangle relative to its frame rectangle.

Modifying the Coordinate System

- `translateOriginToPoint:` (page 3237)
  Translates the receiver's coordinate system so that its origin moves to a new location.
- `scaleUnitSquareToSize:` (page 3208)
  Scales the receiver's coordinate system so that the unit square scales to the specified dimensions.
- `rotateByAngle:` (page 3208)
  Rotates the receiver's bounds rectangle by a specified degree value around the origin of the coordinate system, (0.0, 0.0).

Examining Coordinate System Modifications

- `isFlipped` (page 3181)
  Returns `YES` if the receiver uses flipped drawing coordinates or `NO` if it uses native coordinates.
- `isRotatedFromBase` (page 3184)
  Returns `YES` if the receiver or any of its ancestors has ever received a `setFrameRotation:` (page 3221) or `setBoundsRotation:` (page 3215) message; otherwise returns `NO`.
- `isRotatedOrScaledFromBase` (page 3184)
  Returns `YES` if the receiver or any of its ancestors has ever had a nonzero frame or bounds rotation, or has been scaled from the window's base coordinate system; otherwise returns `NO`.

Base Coordinate Conversion

- `convertPointToBase:` (page 3157)
  Converts the point from the receiver's coordinate system to the base coordinate system.
- `convertPointFromBase:` (page 3156)
  Converts the point from the base coordinate system to the receiver's coordinate system.
- `convertSizeToBase:` (page 3161)
  Converts the size from the receiver's coordinate system to the base coordinate system.
- `convertSizeFromBase:` (page 3161)
  Converts the size from the base coordinate system to the receiver's coordinate system.

- `convertRectToBase:` (page 3159)
  Converts the rectangle from the receiver's coordinate system to the base coordinate system.
- `convertRectFromBase:` (page 3158)
  Converts the rectangle from the base coordinate system to the receiver's coordinate system.

Converting Coordinates

- `convertPoint:fromView:` (page 3155)
  Converts a point from the coordinate system of a given view to that of the receiver.
- `convertPoint:toView:` (page 3155)
  Converts a point from the receiver's coordinate system to that of a given view.
- `convertSize:fromView:` (page 3159)
  Converts a size from another view's coordinate system to that of the receiver.
- `convertSize:toView:` (page 3160)
  Converts a size from the receiver's coordinate system to that of another view.
- `convertRect:fromView:` (page 3157)
  Converts a rectangle from the coordinate system of another view to that of the receiver.
- `convertRect:toView:` (page 3158)
  Converts a rectangle from the receiver's coordinate system to that of another view.
- `centerScanRect:` (page 3153)
  Converts the corners of a specified rectangle to lie on the center of device pixels, which is useful in compensating for rendering overscanning when the coordinate system has been scaled.

Controlling Notifications

- `setPostsFrameChangedNotifications:` (page 3227)
  Controls whether the receiver informs observers when its frame rectangle changes.
- `postsFrameChangedNotifications` (page 3196)
  Returns `YES` if the receiver posts notifications to the default notification center whenever its frame rectangle changes; returns `NO` otherwise.
- `setPostsBoundsChangedNotifications:` (page 3226)
  Controls whether the receiver informs observers when its bounds rectangle changes.
- `postsBoundsChangedNotifications` (page 3196)
  Returns `YES` if the receiver posts notifications to the default notification center whenever its bounds rectangle changes; returns `NO` otherwise.

Resizing Subviews

- `resizeSubviewsWithOldSize:` (page 3206)
  Informs the receiver's subviews that the receiver's bounds rectangle size has changed.
- `resizeWithOldSuperviewSize:` (page 3207)
  Informs the receiver that the bounds size of its superview has changed.

- `setAutoresizesSubviews:` (page 3212)
  Determines whether the receiver automatically resizes its subviews when its frame size changes.
- `autoresizesSubviews` (page 3147)
  Returns `YES` if the receiver automatically resizes its subviews using `resizeSubviewsWithOldSize:` (page 3206) whenever its frame size changes, `NO` otherwise.
- `setAutoresizingMask:` (page 3212)
  Determines how the receiver's `resizeWithOldSuperviewSize:` (page 3207) method changes its frame rectangle.
- `autoresizingMask` (page 3147)
  Returns the receiver's autoresizing mask, which determines how it's resized by the `resizeWithOldSuperviewSize:` (page 3207) method.

Focusing

- `lockFocus` (page 3187)
  Locks the focus on the receiver, so subsequent commands take effect in the receiver's window and coordinate system.
- `lockFocusIfCanDraw` (page 3188)
  Locks the focus to the receiver atomically if the `canDraw` method returns `YES` and returns the value of `canDraw`.
- `lockFocusIfCanDrawInContext:` (page 3188)
  Locks the focus to the receiver atomically if drawing can occur in the specified graphics context.
- `unlockFocus` (page 3239)
  Balances an earlier `lockFocus` (page 3187) or `lockFocusIfCanDraw` (page 3188) message; restoring the focus to the previously focused view is necessary.
- `+ focusView` (page 3137)
  Returns the currently focused `NSView` object, or `nil` if there is none.

Displaying

- `setNeedsDisplay:` (page 3225)
  Controls whether the receiver's entire bounds is marked as needing display.
- `setNeedsDisplayInRect:` (page 3225)
  Marks the region of the receiver within the specified rectangle as needing display, increasing the receiver's existing invalid region to include it.
- `needsDisplay` (page 3191)
  Returns `YES` if the receiver needs to be displayed, as indicated using `setNeedsDisplay:` (page 3225) and `setNeedsDisplayInRect:` (page 3225); returns `NO` otherwise.
- `display` (page 3163)
  Displays the receiver and all its subviews if possible, invoking each the `NSView` methods `lockFocus` (page 3187), `drawRect:` (page 3170), and `unlockFocus` (page 3239) as necessary.
- `displayRect:` (page 3166)
  Acts as `display` (page 3163), but confining drawing to a rectangular region of the receiver.

- `displayRectIgnoringOpacity:` (page 3166)
  Displays the receiver but confines drawing to a specified region and does not back up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.
- `displayRectIgnoringOpacity:inContext:` (page 3166)
  Causes the receiver and its descendants to be redrawn to the specified graphics context.
- `displayIfNeeded` (page 3164)
  Displays the receiver and all its subviews if any part of the receiver has been marked as needing display with a `setNeedsDisplay:` (page 3225) or `setNeedsDisplayInRect:` (page 3225) message.
- `displayIfNeededInRect:` (page 3165)
  Acts as `displayIfNeeded` (page 3164), confining drawing to a specified region of the receiver..
- `displayIfNeededIgnoringOpacity` (page 3165)
  Acts as `displayIfNeeded` (page 3164), except that this method doesn't back up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.
- `displayIfNeededInRectIgnoringOpacity:` (page 3165)
  Acts as `displayIfNeeded` (page 3164), but confining drawing to *aRect* and not backing up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.
- `translateRectsNeedingDisplayInRect:by:` (page 3238)
  Translates the display rectangles by the specified delta.
- `isOpaque` (page 3183)
  Overridden by subclasses to return `YES` if the receiver is opaque, `NO` otherwise.
- `viewWillDraw` (page 3242)
  Informs the receiver that it will be required to draw content.

Focus Ring Drawing

- `setKeyboardFocusRingNeedsDisplayInRect:` (page 3223)
  Invalidates the area around the focus ring.
+ `defaultFocusRingType` (page 3137)
  Returns the default focus ring type.
- `setFocusRingType:` (page 3218)
  Sets the type of focus ring to be drawn around the receiver.
- `focusRingType` (page 3175)
  Returns the type of focus ring drawn around the receiver.

Fullscreen Mode

- `enterFullScreenMode:withOptions:` (page 3173)
  Sets the receiver to full screen mode.
- `exitFullScreenModeWithOptions:` (page 3174)
  Instructs the receiver to exit full screen mode.
- `isInFullScreenMode` (page 3183)
  Returns whether the view is in full screen mode.

Hiding Views

- `setHidden:` (page 3222)
  Sets whether the view is hidden.
- `isHidden` (page 3182)
  Returns whether the receiver is marked as hidden.
- `isHiddenOrHasHiddenAncestor` (page 3182)
  Returns `YES` if the receiver is marked as hidden or has an ancestor in the view hierarchy that is marked as hidden; returns `NO` otherwise.
- `viewDidHide` (page 3240)
  Invoked when the receiver is hidden, either directly, or in response to an ancestor being hidden.
- `viewDidUnhide` (page 3242)
  Invoked when the receiver is unhidden, either directly, or in response to an ancestor being unhidden

Drawing

- `canDrawConcurrently` (page 3153)
  Returns whether the view's `drawRect:` method can be invoked on a background thread.
- `setCanDrawConcurrently:` (page 3217)
  Sets whether the view's `drawRect:` method can be invoked on a background thread.
- `drawRect:` (page 3170)
  Overridden by subclasses to draw the receiver's image within the passed-in rectangle.
- `visibleRect` (page 3245)
  Returns the portion of the receiver not clipped by its superviews.
- `canDraw` (page 3152)
  Returns `YES` if drawing commands will produce any result, `NO` otherwise.
- `shouldDrawColor` (page 3232)
  Returns `NO` if the receiver is being drawn in an `NSWindow` object (as opposed, for example, to being printed) and the window object can't store color; otherwise returns `YES`.
- `getRectsBeingDrawn:count:` (page 3176)
  Returns by indirection a list of non-overlapping rectangles that define the area the receiver is being asked to draw in `drawRect:` (page 3170).
- `needsToDrawRect:` (page 3192)
  Returns whether the specified rectangle intersects any part of the area that the receiver is being asked to draw.
- `wantsDefaultClipping` (page 3246)
  Returns whether the Application Kit's default clipping provided to `drawRect:` (page 3170) implementations is in effect.
- `bitmapImageRepForCachingDisplayInRect:` (page 3150)
  Returns a bitmap-representation object suitable for caching the specified portion of the receiver.
- `cacheDisplayInRect:toBitmapImageRep:` (page 3151)
  Draws the specified area of the receiver, and its descendants, into a provided bitmap-representation object.

Managing Live Resize

- `inLiveResize` (page 3180)
  A convenience method, expected to be called from `drawRect:` (page 3170), to assist in decisions about optimized drawing.
- `preservesContentDuringLiveResize` (page 3196)
  Returns `YES` if the view supports the optimization of live-resize operations by preserving content that has not moved; otherwise, returns `NO`.
- `getRectsExposedDuringLiveResize:count:` (page 3177)
  Returns a list of rectangles indicating the newly exposed areas of the receiver.
- `rectPreservedDuringLiveResize` (page 3199)
  Returns the rectangle identifying the portion of your view that did not change during a live resize operation.
- `viewWillStartLiveResize` (page 3244)
  Informs the receiver of the start of a live resize.
- `viewDidEndLiveResize` (page 3240)
  Informs the receiver of the end of a live resize.

Managing the Graphics State

- `allocateGState` (page 3146)
  Causes the receiver to maintain a private graphics state object, which encapsulates all parameters of the graphics environment.
- `gState` (page 3178)
  Returns the identifier for the receiver's graphics state object, or 0 if the receiver doesn't have a graphics state object.
- `setUpGState` (page 3229)
  Overridden by subclasses to (re)initialize the receiver's graphics state object.
- `renewGState` (page 3205)
  Invalidates the receiver's graphics state object, if it has one.
- `releaseGState` (page 3201)
  Frees the receiver's graphics state object, if it has one.

Event Handling

- `acceptsFirstMouse:` (page 3138)
  Overridden by subclasses to return `YES` if the receiver should be sent a `mouseDown:` (page 2164) message for an initial mouse-down event, `NO` if not.
- `hitTest:` (page 3179)
  Returns the farthest descendant of the receiver in the view hierarchy (including itself) that contains a specified point, or `nil` if that point lies completely outside the receiver.
- `mouse:inRect:` (page 3189)
  Returns whether a region of the receiver contains a specified point, accounting for whether the receiver is flipped or not.

- `performKeyEquivalent:` (page 3195)
  Implemented by subclasses to respond to key equivalents (also known as keyboard shortcuts).
- `rightMouseDown:` (page 3207)
  Informs the receiver that the user has pressed the right mouse button.
- `performMnemonic:` (page 3195)
  Implemented by subclasses to respond to mnemonics.
- `mouseDownCanMoveWindow` (page 3190)
  Returns `YES` if the receiver does not need to handle a mouse down and can pass it through to superviews; `NO` if it needs to handle the mouse down.
- `inputContext` (page 3181)
  Returns the text input context object for the receiver.

Dragging Operations

- `dragImage:at:offset:event:pasteboard:source:slideBack:` (page 3168)
  Initiates a dragging operation from the receiver, allowing the user to drag arbitrary data with a specified icon into any application that has window or view objects that accept dragged data.
- `dragFile:fromRect:slideBack:event:` (page 3167)
  Initiates a dragging operation from the receiver, allowing the user to drag a file icon to any application that has window or view objects that accept files.
- `registerForDraggedTypes:` (page 3201)
  Registers the pasteboard types that the receiver will accept as the destination of an image-dragging session.
- `registeredDraggedTypes` (page 3200)
  Returns the array of pasteboard drag types that the view can accept.
- `unregisterDraggedTypes` (page 3239)
  Unregisters the receiver as a possible destination in a dragging session.
- `shouldDelayWindowOrderingForEvent:` (page 3231)
  Overridden by subclasses to allow the user to drag images from the receiver without its window moving forward and possibly obscuring the destination and without activating the application.
- `dragPromisedFilesOfTypes:fromRect:source:slideBack:event:` (page 3169)
  Initiates a dragging operation from the receiver, allowing the user to drag one or more promised files (or directories) into any application that has window or view objects that accept promised file data.

Tool Tips

- `addToolTipRect:owner:userData:` (page 3141)
  Creates a tool tip for a defined area the receiver and returns a tag that identifies the tool tip rectangle.
- `removeAllToolTips` (page 3201)
  Removes all tool tips assigned to the receiver.
- `removeToolTip:` (page 3203)
  Removes the tool tip identified by specified tag.
- `setToolTip:` (page 3229)
  Sets the tool tip text for the view to *`string`*.

- `toolTip` (page 3236)
  Returns the text for the view's tool tip.

Managing Tracking Rectangles

- `addTrackingRect:owner:userData:assumeInside:` (page 3142)
  Establishes an area for tracking mouse-entered and mouse-exited events within the receiver and returns a tag that identifies the tracking rectangle.
- `removeTrackingRect:` (page 3204)
  Removes the tracking rectangle identified by a tag.

Managing Tracking Areas

- `addTrackingArea:` (page 3142)
  Adds a given tracking area to the receiver.
- `removeTrackingArea:` (page 3204)
  Removes a given tracking area from the receiver.
- `trackingAreas` (page 3237)
  Returns an array of the receiver's tracking areas.
- `updateTrackingAreas` (page 3239)
  Invoked automatically when the view's geometry changes such that its tracking areas need to be recalculated.

Managing Cursor Tracking

- `addCursorRect:cursor:` (page 3139)
  Establishes the cursor to be used when the mouse pointer lies within a specified region.
- `removeCursorRect:cursor:` (page 3202)
  Completely removes a cursor rectangle from the receiver.
- `discardCursorRects` (page 3163)
  Invalidates all cursor rectangles set up using `addCursorRect:cursor:` (page 3139).
- `resetCursorRects` (page 3206)
  Overridden by subclasses to define their default cursor rectangles.

Scrolling

- `scrollPoint:` (page 3209)
  Scrolls the receiver's closest ancestor `NSClipView` object so a point in the receiver lies at the origin of the clip view's bounds rectangle.
- `scrollRectToVisible:` (page 3210)
  Scrolls the receiver's closest ancestor `NSClipView` object the minimum distance needed so a specified region of the receiver becomes visible in the clip view.

- `autoscroll:` (page 3148)
  Scrolls the receiver's closest ancestor `NSClipView` object proportionally to the distance of an event that occurs outside of it.
- `adjustScroll:` (page 3145)
  Overridden by subclasses to modify a given rectangle, returning the altered rectangle.
- `scrollRect:by:` (page 3210)
  Copies the visible portion of the receiver's rendered image within a region and lays that portion down again at a specified offset .
- `enclosingScrollView` (page 3172)
  Returns the nearest ancestor `NSScrollView` object containing the receiver (not including the receiver itself); otherwise returns `nil`.
- `scrollClipView:toPoint:` (page 3209)
  Notifies the superview of a clip view that the clip view needs to reset the origin of its bounds rectangle.
- `reflectScrolledClipView:` (page 3200)
  Notifies a clip view's superview that either the clip view's bounds rectangle or the document view's frame rectangle has changed, and that any indicators of the scroll position need to be adjusted.

Contextual Menus

- `menuForEvent:` (page 3189)
  Overridden by subclasses to return a context-sensitive pop-up menu for a given mouse-down event.
+ `defaultMenu` (page 3137)
  Overridden by subclasses to return the default pop-up menu for instances of the receiving class.

Key-view Loop Management

- `canBecomeKeyView` (page 3152)
  Returns whether the receiver can become key view.
- `needsPanelToBecomeKey` (page 3191)
  Overridden by subclasses to determine if the receiver requires its panel, which might otherwise avoid becoming key, to become the key window so that it can handle keyboard input and navigation.
- `setNextKeyView:` (page 3226)
  Inserts a specified view object after the receiver in the key view loop of the receiver's window.
- `nextKeyView` (page 3192)
  Returns the view object following the receiver in the key view loop.
- `nextValidKeyView` (page 3193)
  Returns the closest view object in the key view loop that follows the receiver and accepts first responder status.
- `previousKeyView` (page 3197)
  Returns the view object preceding the receiver in the key view loop.
- `previousValidKeyView` (page 3197)
  Returns the closest view object in the key view loop that precedes the receiver and accepts first responder status.

Printing

- `print:` (page 3198)
  This action method opens the Print panel, and if the user chooses an option other than canceling, prints the receiver and all its subviews to the device specified in the Print panel.
- `beginPageInRect:atPlacement:` (page 3149)
  Called at the beginning of each page, this method sets up the coordinate system so that a region inside the receiver's bounds is translated to a specified location..
- `dataWithEPSInsideRect:` (page 3162)
  Returns EPS data that draws the region of the receiver within a specified rectangle.
- `dataWithPDFInsideRect:` (page 3162)
  Returns PDF data that draws the region of the receiver within a specified rectangle.
- `printJobTitle` (page 3198)
  Returns the receiver's print job title.
- `pageFooter` (page 3194)
  Returns a default footer string that includes the current page number and page count.
- `pageHeader` (page 3194)
  Returns a default header string that includes the print job title and date.
- `writeEPSInsideRect:toPasteboard:` (page 3248)
  Writes EPS data that draws the region of the receiver within a specified rectangle onto a pasteboard.
- `writePDFInsideRect:toPasteboard:` (page 3249)
  Writes PDF data that draws the region of the receiver within a specified rectangle onto a pasteboard.
- `drawPageBorderWithSize:` (page 3170)
  Allows applications that use the Application Kit pagination facility to draw additional marks on each logical page.
- `drawSheetBorderWithSize:` (page 3171)
  Allows applications that use the Application Kit pagination facility to draw additional marks on each printed sheet.

Pagination

- `heightAdjustLimit` (page 3178)
  Returns the fraction (from 0.0 to 1.0) of the page that can be pushed onto the next page during automatic pagination to prevent items such as lines of text from being divided across pages.
- `widthAdjustLimit` (page 3247)
  Returns the fraction (from 0.0 to 1.0) of the page that can be pushed onto the next page during automatic pagination to prevent items such as small images or text columns from being divided across pages.
- `adjustPageWidthNew:left:right:limit:` (page 3144)
  Overridden by subclasses to adjust page width during automatic pagination.
- `adjustPageHeightNew:top:bottom:limit:` (page 3143)
  Overridden by subclasses to adjust page height during automatic pagination.
- `knowsPageRange:` (page 3185)
  Returns `YES` if the receiver handles page boundaries, `NO` otherwise.

- `rectForPage:` (page 3199)
  Implemented by subclasses to determine the portion of the receiver to be printed for the page number page.
- `locationOfPrintRect:` (page 3186)
  Invoked by `print:` (page 3198) to determine the location of the region of the receiver being printed on the physical page.

Writing Conforming Rendering Instructions

- `beginDocument` (page 3149)
  Invoked at the beginning of the printing session, this method sets up the current graphics context.
- `endDocument` (page 3173)
  This method is invoked at the end of the printing session.
- `endPage` (page 3173)
  Writes the end of a conforming page.

Core Animation Layer-Backing

- `layer` (page 3185)
  Returns the Core Animation layer that the receiver uses as its backing store.
- `setLayer:` (page 3223)
  Sets the Core Animation layer that the receiver uses for layer-backing to the specified layer.
- `wantsLayer` (page 3246)
  Returns a Boolean value that indicates whether the receiver is using a layer as its backing store.
- `setWantsLayer:` (page 3230)
  Specifies whether the receiver and its subviews use a Core Animation layer as a backing store.
- `makeBackingLayer` (page 3189)
  Creates the view's backing layer.
- `layerContentsPlacement` (page 3186)
  Returns the current layer contents placement policy.
- `setLayerContentsPlacement:` (page 3223)
  Sets the view's layer contents placement policy.
- `layerContentsRedrawPolicy` (page 3186)
  Returns the view's layer contents redraw policy.
- `setLayerContentsRedrawPolicy:` (page 3224)
  Sets the receiver layer contents redraw policy.

Core Animation Layer Properties

- `setFrameCenterRotation:` (page 3219)
  Rotates the frame of the receiver about the layer's position.
- `frameCenterRotation` (page 3176)
  Returns the receiver's rotation about the layer's position.

- `setAlphaValue:` (page 3211)
  Sets the opacity of the receiver.
- `alphaValue` (page 3146)
  Returns the opacity of the receiver
- `setBackgroundFilters:` (page 3213)
  An array of CoreImage filters that are applied to the receiver's background.
- `backgroundFilters` (page 3149)
  Returns the array of CoreImage filters that are applied to the receiver's background
- `setCompositingFilter:` (page 3217)
  Sets a CoreImage filter that is used to composite the receiver's contents with the background.
- `compositingFilter` (page 3154)
  Returns the CoreImage filter that is used to composite the receiver's contents with the background
- `setContentFilters:` (page 3218)
  Sets the array of CoreImage filters that are applied to the contents of the receiver and its sublayers.
- `contentFilters` (page 3154)
  Returns the array of CoreImage filters that are applied to the contents of the receiver and its sublayers.
- `setShadow:` (page 3228)
  Sets the shadow drawn by the receiver.
- `shadow` (page 3231)
  Returns the shadow drawn by the receiver

Displaying Definition Windows

- `showDefinitionForAttributedString:atPoint:` (page 3232)
  Shows a window displaying the definition of the of the attributed string at the specified point.
- `showDefinitionForAttributedString:range:options:baselineOriginProvider:` (page 3233)
  Shows a window displaying the definition of the specified range of the attributed string.

Touch Event Handling

- `acceptsTouchEvents` (page 3138)
  Returns whether the view will accept touch events.
- `setAcceptsTouchEvents:` (page 3211)
  Sets whether the view should accept touch events.
- `wantsRestingTouches` (page 3247)
  Returns whether the view wants resting touches.
- `setWantsRestingTouches:` (page 3230)
  Sets whether the view wants to receive resting touch events.

Class Methods

defaultFocusRingType

Returns the default focus ring type.

```
+ (NSFocusRingType)defaultFocusRingType
```

Return Value
The default type of focus ring for objects of the receiver's class. Possible return values are listed in `NSFocusRingType` (page 4009).

Discussion
If `NSFocusRingTypeDefault` is returned from the instance method `focusRingType` (page 3175), the receiver can invoke this class method to find out what type of focus ring is the default. The receiver is free to ignore the default setting.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSView.h`

defaultMenu

Overridden by subclasses to return the default pop-up menu for instances of the receiving class.

```
+ (NSMenu *)defaultMenu
```

Discussion
The default implementation returns `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- menuForEvent:` (page 3189)
`menu` (page 2163) (`NSResponder`)

Declared In
`NSView.h`

focusView

Returns the currently focused `NSView` object, or `nil` if there is none.

```
+ (NSView *)focusView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lockFocus` (page 3187)
- `unlockFocus` (page 3239)

Declared In
`NSView.h`

Instance Methods

acceptsFirstMouse:

Overridden by subclasses to return `YES` if the receiver should be sent a `mouseDown:` (page 2164) message for an initial mouse-down event, `NO` if not.

```
- (BOOL)acceptsFirstMouse:(NSEvent *)theEvent
```

Parameters
theEvent
The initial mouse-down event, which must be over the receiver in its window.

Discussion
The receiver can either return a value unconditionally or use the location of *theEvent* to determine whether or not it wants the event. The default implementation ignores *theEvent* and returns `NO`.

Override this method in a subclass to allow instances to respond to click-through. This allows the user to click on a view in an inactive window, activating the view with one click, instead of clicking first to make the window active and then clicking the view. Most view objects refuse a click-through attempt, so the event simply activates the window. Many control objects, however, such as instances of `NSButton` and `NSSlider`, do accept them, so the user can immediately manipulate the control without having to release the mouse button.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `hitTest:` (page 3179)

Declared In
`NSView.h`

acceptsTouchEvents

Returns whether the view will accept touch events.

```
- (BOOL)acceptsTouchEvents
```

Return Value
`YES` if the view accepts touch events, otherwise `NO`.

Discussion
The default is `NO`.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setAcceptsTouchEvents:` (page 3211)

Declared In
`NSView.h`

addCursorRect:cursor:

Establishes the cursor to be used when the mouse pointer lies within a specified region.

```
- (void)addCursorRect:(NSRect)aRect
    cursor:(NSCursor *)aCursor
```

Parameters

aRect
A rectangle defining a region of the receiver.

aCursor
An object representing a cursor.

Discussion
Cursor rectangles aren't subject to clipping by superviews, nor are they intended for use with rotated views. You should explicitly confine a cursor rectangle to the view's visible rectangle to prevent improper behavior.

This method is intended to be invoked only by the `resetCursorRects` (page 3206) method. If invoked in any other way, the resulting cursor rectangle will be discarded the next time the view's cursor rectangles are rebuilt.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `removeCursorRect:cursor:` (page 3202)
- `discardCursorRects` (page 3163)
- `resetCursorRects` (page 3206)
- `visibleRect` (page 3245)

Related Sample Code
DragItemAround

Declared In
`NSView.h`

addSubview:

Adds a view to the receiver's subviews so it's displayed above its siblings.

```
- (void)addSubview:(NSView *)aView
```

Parameters

aView

The view to add to the receiver as a subview.

Discussion

This method also sets the receiver as the next responder of *aView*.

The receiver retains *aView*. If you use `removeFromSuperview` (page 3202) to remove *aView* from the view hierarchy, *aView* is released. If you want to keep using *aView* after removing it from the view hierarchy (if, for example, you are swapping through a number of views), you must retain it before invoking `removeFromSuperview` (page 3202).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addSubview:positioned:relativeTo:` (page 3140)
- `subviews` (page 3235)
- `removeFromSuperview` (page 3202)

`setNextResponder:` (page 2197) (NSResponder)

- `viewWillMoveToSuperview:` (page 3243)
- `viewWillMoveToWindow:` (page 3243)

Related Sample Code

FunHouse

GLUT

MenuMadness

Quartz Composer WWDC 2005 TextEdit

WhackedTV

Declared In

`NSView.h`

addSubview:positioned:relativeTo:

Inserts a view among the receiver's subviews so it's displayed immediately above or below another view.

```
- (void)addSubview:(NSView *)aView
    positioned:(NSWindowOrderingMode)place
    relativeTo:(NSView *)otherView
```

Parameters

aView

The view object to add to the receiver as a subview.

place

An `enum` constant specifying the position of the `aView` relative to `otherView`. Valid values are `NSWindowAbove` or `NSWindowBelow`.

otherView

The other view `aView` is to be positioned relative to. If *otherView* is `nil` (or isn't a subview of the receiver), *aView* is added above or below all of its new siblings.

Discussion
This method also sets the receiver as the next responder of *aView*.

The receiver retains *aView*. If you use `removeFromSuperview` (page 3202) to remove *aView* from the view hierarchy, *aView* is released. If you want to keep using *aView* after removing it from the view hierarchy (if, for example, you are swapping through a number of views), you must retain it before invoking `removeFromSuperview` (page 3202).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addSubview:` (page 3139)
- `subviews` (page 3235)
- `removeFromSuperview` (page 3202)

`setNextResponder:` (page 2197) (`NSResponder`)

Related Sample Code
GLUT

Declared In
`NSView.h`

addToolTipRect:owner:userData:

Creates a tool tip for a defined area the receiver and returns a tag that identifies the tool tip rectangle.

```
- (NSToolTipTag)addToolTipRect:(NSRect)aRect
    owner:(id)anObject
    userData:(void *)userData
```

Parameters

aRect

A rectangle defining the region of the receiver to associate the tool tip with.

anObject

An object from which to obtain the tool tip string. The object should either implement `view:stringForToolTip:point:userData:` (page 3921), or return a suitable string from its `description` method. (It can therefore simply be an `NSString` object.)

Important: The receiver maintains a weak reference to *anObject*. You are responsible for ensuring that *anObject* remains valid for as long as it may be needed.

userData

Any additional information you want to pass to `view:stringForToolTip:point:userData:` (page 3921); it is not used if *anObject* does not implement this method.

Return Value
An integer tag identifying the tool tip; you can use this tag to remove the tool tip.

Discussion
The tool tip string is obtained dynamically from *anObject* by invoking either the `NSToolTipOwner` informal protocol method `view:stringForToolTip:point:userData:` (page 3921), if implemented, or the `NSObject` protocol method `description`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `removeToolTip:` (page 3203)
- `removeAllToolTips` (page 3201)

Declared In
`NSView.h`

addTrackingArea:

Adds a given tracking area to the receiver.

```
- (void)addTrackingArea:(NSTrackingArea *)trackingArea
```

Parameters
trackingArea
The tracking area to add to the receiver.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
BasicCocoaAnimations
MenuItemView
PhotoSearch
TrackIt

Declared In
`NSView.h`

addTrackingRect:owner:userData:assumeInside:

Establishes an area for tracking mouse-entered and mouse-exited events within the receiver and returns a tag that identifies the tracking rectangle.

```
- (NSTrackingRectTag)addTrackingRect:(NSRect)aRect
    owner:(id)userObject
    userData:(void *)userData
    assumeInside:(BOOL)flag
```

Parameters
aRect
A rectangle that defines a region of the receiver for tracking mouse-entered and mouse-exited events.

userObject

The object that gets sent the event messages. It can be the receiver itself or some other object (such as an NSCursor or a custom drawing tool object), as long as it responds to both `mouseEntered:` (page 2165) and `mouseExited:` (page 2165).

userData

Data stored in the `NSEvent` object for each tracking event.

flag

If `YES`, the first event will be generated when the cursor leaves *aRect*, regardless if the cursor is inside *aRect* when the tracking rectangle is added. If `NO` the first event will be generated when the cursor leaves *aRect* if the cursor is initially inside *aRect*, or when the cursor enters *aRect* if the cursor is initially outside *aRect*. You usually want to set this flag to `NO`.

Return Value
A tag that identifies the tracking rectangle. It is stored in the associated `NSEvent` objects and can be used to remove the tracking rectangle.

Discussion
Tracking rectangles provide a general mechanism that can be used to trigger actions based on the cursor location (for example, a status bar or hint field that provides information on the item the cursor lies over). To simply change the cursor over a particular area, use `addCursorRect:cursor:` (page 3139). If you must use tracking rectangles to change the cursor, the `NSCursor` class specification describes the additional methods that must be invoked to change cursors by using tracking rectangles.

On Mac OS X v10.5 and later, tracking areas provide a greater range of functionality (see `addTrackingArea:` (page 3142)).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- removeTrackingRect:` (page 3204)
`- addTrackingArea:` (page 3142)
`userData` (page 1089) (`NSEvent`)

Related Sample Code
FunHouse
FunkyOverlayWindow
GLUT

Declared In
`NSView.h`

adjustPageHeightNew:top:bottom:limit:

Overridden by subclasses to adjust page height during automatic pagination.

```
- (void)adjustPageHeightNew:(CGFloat *)newBottom
    top:(CGFloat)top
    bottom:(CGFloat)proposedBottom
    limit:(CGFloat)bottomLimit
```

Parameters

newBottom

Returns by indirection a new `float` value for the bottom edge of the pending page rectangle in the receiver's coordinate system.

top

A `float` value that sets the top edge of the pending page rectangle in the receiver's coordinate system.

proposedBottom

A `float` value that sets the bottom edge of the pending page rectangle in the receiver's coordinate system.

bottomLimit

The topmost float value *newBottom* can be set to, as calculated using the return value of `heightAdjustLimit` (page 3178).

Discussion

This method is invoked by `print:` (page 3198). The receiver can raise the bottom edge and return the new value in *newBottom*, allowing it to prevent items such as lines of text from being divided across pages. If *bottomLimit* is exceeded, the pagination mechanism simply uses *bottomLimit* for the bottom edge.

The default implementation of this method propagates the message to its subviews, allowing nested views to adjust page height for their drawing as well. An `NSButton` object or other small view, for example, will nudge the bottom edge up if necessary to prevent itself from being cut in two (thereby pushing it onto an adjacent page). Subclasses should invoke `super`'s implementation, if desired, after first making their own adjustments.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- adjustPageWidthNew:left:right:limit:` (page 3144)

Declared In

`NSView.h`

adjustPageWidthNew:left:right:limit:

Overridden by subclasses to adjust page width during automatic pagination.

```
- (void)adjustPageWidthNew:(CGFloat *)newRight
    left:(CGFloat)left
    right:(CGFloat)proposedRight
    limit:(CGFloat)rightLimit
```

Parameters

newRight

Returns by indirection a new `float` value for the right edge of the pending page rectangle in the receiver's coordinate system.

left

A `float` value that sets the left edge of the pending page rectangle in the receiver's coordinate system.

proposedRight

A `float` value that sets the right edge of the pending page rectangle in the receiver's coordinate system.

rightLimit

The leftmost `float` value *newRight* can be set to, as calculated using the return value of `widthAdjustLimit` (page 3247).

Discussion
This method is invoked by `print:` (page 3198). The receiver can pull in the right edge and return the new value in *newRight*, allowing it to prevent items such as small images or text columns from being divided across pages. If *rightLimit* is exceeded, the pagination mechanism simply uses *rightLimit* for the right edge.

The default implementation of this method propagates the message to its subviews, allowing nested views to adjust page width for their drawing as well. An `NSButton` object or other small view, for example, will nudge the right edge out if necessary to prevent itself from being cut in two (thereby pushing it onto an adjacent page). Subclasses should invoke `super`'s implementation, if desired, after first making their own adjustments.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- adjustPageHeightNew:top:bottom:limit:` (page 3143)

Declared In
`NSView.h`

adjustScroll:

Overridden by subclasses to modify a given rectangle, returning the altered rectangle.

```
- (NSRect)adjustScroll:(NSRect)proposedVisibleRect
```

Parameters

proposedVisibleRect

A rectangle defining a region of the receiver.

Discussion
`NSClipView` invokes this method to allow its document view to adjust its position during scrolling. For example, a custom view object that displays a table of data can adjust the origin of *proposedVisibleRect* so rows or columns aren't cut off by the edge of the enclosing `NSClipView`. `NSView`'s implementation simply returns *proposedVisibleRect*.

`NSClipView` only invokes this method during automatic or user controlled scrolling. Its `scrollToPoint:` (page 635) method doesn't invoke this method, so you can still force a scroll to an arbitrary point.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

allocateGState

Causes the receiver to maintain a private graphics state object, which encapsulates all parameters of the graphics environment.

```
- (void)allocateGState
```

Discussion
If you do not invoke `allocateGState`, a graphics state object is constructed from scratch each time the NSView is focused.

The receiver builds the graphics state parameters using `setUpGState` (page 3229), then automatically establishes this graphics state each time the focus is locked on it. A graphics state may improve performance for view objects that are focused often and need to set many parameters, but use of standard rendering operators is normally efficient enough.

Because graphics states occupy a fair amount of memory, they can actually degrade performance. Be sure to test application performance with and without the private graphics state before committing to its use.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setUpGState` (page 3229)
- `gState` (page 3178)
- `renewGState` (page 3205)
- `releaseGState` (page 3201)
- `lockFocus` (page 3187)

Related Sample Code
GLUT

Declared In
`NSView.h`

alphaValue

Returns the opacity of the receiver

```
- (CGFloat)alphaValue
```

Return Value
The current opacity of the receiver

Discussion
This method returns the value of the `opacity` property of the receiver's layer. Possible values are between 0.0 (transparent) and 1.0 (opaque). The default is 1.0.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
RoundTransparentWindow

Declared In
NSView.h

ancestorSharedWithView:

Returns the closest ancestor shared by the receiver and a given view.

```
- (NSView *)ancestorSharedWithView:(NSView *)aView
```

Parameters

aView

The view to test (along with the receiver) for closest shared ancestor.

Return Value
The closest ancestor or `nil` if there's no such object. Returns `self` if *aView* is identical to the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isDescendantOf:` (page 3181)

Declared In
NSView.h

autoresizesSubviews

Returns `YES` if the receiver automatically resizes its subviews using `resizeSubviewsWithOldSize:` (page 3206) whenever its frame size changes, `NO` otherwise.

```
- (BOOL)autoresizesSubviews
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAutoresizesSubviews:` (page 3212)

Declared In
NSView.h

autoresizingMask

Returns the receiver's autoresizing mask, which determines how it's resized by the `resizeWithOldSuperviewSize:` (page 3207) method.

```
- (NSUInteger)autoresizingMask
```

Return Value
An integer bit mask specified by combining using the C bitwise OR operator any of the options described in "Resizing masks" (page 3250).

Discussion
If the autoresizing mask is equal to `NSViewNotSizable` (that is, if none of the options are set), then the receiver doesn't resize at all in `resizeWithOldSuperviewSize:` (page 3207).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
PhotoSearch
PredicateEditorSample

Declared In
`NSView.h`

autoscroll:

Scrolls the receiver's closest ancestor `NSClipView` object proportionally to the distance of an event that occurs outside of it.

```
- (BOOL)autoscroll:(NSEvent *)theEvent
```

Parameters
theEvent
An event object whose location should be expressed in the window's base coordinate system (which it normally is), not the receiving view's.

Return Value
Returns `YES` if any scrolling is performed; otherwise returns `NO`.

Discussion
View objects that track mouse-dragged events can use this method to scroll automatically when the cursor is dragged outside of the `NSClipView` object. Repeated invocations of this method (with an appropriate delay) result in continual scrolling, even when the mouse doesn't move.

Availability
Available in Mac OS X v10.0 and later.

See Also
`autoscroll:` (page 632) (NSClipView)
`- scrollPoint:` (page 3209)
`- isDescendantOf:` (page 3181)

Related Sample Code
DragItemAround
Rulers

Declared In
`NSView.h`

backgroundFilters

Returns the array of CoreImage filters that are applied to the receiver's background

```
- (NSArray *)backgroundFilters
```

Return Value
An array of CoreImage filters.

Discussion
This method returns the value of the `backgroundFilters` property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

beginDocument

Invoked at the beginning of the printing session, this method sets up the current graphics context.

```
- (void)beginDocument
```

Discussion
Note that this method may be invoked in a subthread.

Override it to configure printing related settings. You should store your settings in the object returned by `NSPrintInfo`'s `sharedPrintInfo` (page 2048) class method, which is guaranteed to return an instance specific to the thread in which you invoke this method. If you override this method, call the superclass implementation.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

beginPageInRect:atPlacement:

Called at the beginning of each page, this method sets up the coordinate system so that a region inside the receiver's bounds is translated to a specified location..

```
- (void)beginPageInRect:(NSRect)aRect
    atPlacement:(NSPoint)location
```

Parameters

aRect
A rectangle defining the region to be translated.

location
A point that is the end-point of translation.

Discussion
If you override this method, be sure to call the superclass implementation.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

bitmapImageRepForCachingDisplayInRect:

Returns a bitmap-representation object suitable for caching the specified portion of the receiver.

```
- (NSBitmapImageRep *)bitmapImageRepForCachingDisplayInRect:(NSRect)aRect
```

Parameters

aRect

A rectangle defining the area of the receiver to be cached.

Return Value
An autoreleased `NSBitmapImageRep` object or `nil` if the object could not be created.

Discussion
Passing the visible rectangle of the receiver (`[self visibleRect]`) returns a bitmap suitable for caching the current contents of the view, including all of its descendants.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- cacheDisplayInRect:toBitmapImageRep:` (page 3151)

Related Sample Code
AnimatedTableView
Reducer

Declared In
`NSView.h`

bounds

Returns the receiver's bounds rectangle, which expresses its location and size in its own coordinate system.

```
- (NSRect)bounds
```

Discussion
By default, the origin of the returned rectangle is (0, 0) and its size matches the size of the receiver's frame rectangle (measured in points). In Mac OS X v10.5 and later, if the receiver is being rendered into an OpenGL graphics context (using an `NSOpenGLContext` object), the default bounds origin is still (0, 0) but the default bounds size is measured in pixels instead of points. Thus, for user space scale factors other than 1.0, the default size of the bounds rectangle may be bigger or smaller than the default size of the frame rectangle when drawing with OpenGL.

Important: Developers of OpenGL applications should not rely on this method converting coordinates to pixels automatically in future releases. Instead, you should convert coordinates to device space explicitly using the `convertPointToBase:` (page 3157), `convertSizeToBase:` (page 3161), or `convertRectToBase:` (page 3159) methods or their earlier counterparts `convertPoint:toView:` (page 3155), `convertSize:toView:` (page 3160), or `convertRect:toView:` (page 3158).

If you explicitly change the origin or size of the bounds rectangle, this method does not return the default rectangle and instead returns the rectangle you set. If you add a rotation factor to the view, however, that factor is also reflected in the returned bounds rectangle. You can determine if a rotation factor is in effect by calling the `boundsRotation` (page 3151) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `frame` (page 3175)
- `setBounds:` (page 3214)

Related Sample Code
From A View to A Movie
FunHouse
GLUT
TargetGallery
WhackedTV

Declared In
`NSView.h`

boundsRotation

Returns the angle, in degrees, of the receiver's bounds rectangle relative to its frame rectangle.

```
- (CGFloat)boundsRotation
```

Discussion
See the `setBoundsRotation:` (page 3215) method description for more information on bounds rotation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `rotateByAngle:` (page 3208)
- `setBoundsRotation:` (page 3215)

Declared In
`NSView.h`

cacheDisplayInRect:toBitmapImageRep:

Draws the specified area of the receiver, and its descendants, into a provided bitmap-representation object.

```
- (void)cacheDisplayInRect:(NSRect)rect
    toBitmapImageRep:(NSBitmapImageRep *)bitmapImageRep
```

Parameters

rect

A rectangle defining the region to be drawn into *bimapImageRep*.

bitmapImageRep

An `NSBitmapImageRep` object. For pixel-format compatibility, *bitmapImageRep* should have been obtained from `bitmapImageRepForCachingDisplayInRect:` (page 3150).

Discussion

You are responsible for initializing the bitmap to the desired configuration before calling this method. However, once initialized, you can reuse the same bitmap multiple times to refresh the cached copy of your view's contents.

The bitmap produced by this method is transparent (that is, has an alpha value of 0) wherever the receiver and its descendants do not draw any content.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- bitmapImageRepForCachingDisplayInRect:` (page 3150)

Related Sample Code

AnimatedTableView

Reducer

Declared In

`NSView.h`

canBecomeKeyView

Returns whether the receiver can become key view.

```
- (BOOL)canBecomeKeyView
```

Return Value

Returns `YES` if the receiver can become key view, `NO` otherwise.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`NSView.h`

canDraw

Returns `YES` if drawing commands will produce any result, `NO` otherwise.

```
- (BOOL)canDraw
```

Discussion
Use this method when invoking a draw method directly along with `lockFocus` (page 3187) and `unlockFocus` (page 3239), bypassing the `display...` methods (which test drawing ability and perform locking for you). If this method returns `NO`, you shouldn't invoke `lockFocus` (page 3187) or perform any drawing.

A view object can draw on-screen if it is not hidden, it is attached to a view hierarchy in a window (`NSWindow`), and the window has a corresponding window device. A view object can draw during printing if it is a descendant of the view being printed.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setHidden:` (page 3222)

Declared In
`NSView.h`

canDrawConcurrently

Returns whether the view's `drawRect:` method can be invoked on a background thread.

```
- (BOOL)canDrawConcurrently
```

Return Value
`YES` if `drawRect:` (page 3170) can be invoked from a background thread, otherwise `NO`. The default is `NO`.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setCanDrawConcurrently:` (page 3217)

Declared In
`NSView.h`

centerScanRect:

Converts the corners of a specified rectangle to lie on the center of device pixels, which is useful in compensating for rendering overscanning when the coordinate system has been scaled.

```
- (NSRect)centerScanRect:(NSRect)aRect
```

Parameters
aRect
The rectangle whose corners are to be converted.

Return Value
The adjusted rectangle.

Discussion
This method converts the given rectangle to device coordinates, adjusts the rectangle to lie in the center of the pixels, and converts the resulting rectangle back to the receiver's coordinate system. Note that this method does not take into account any transformations performed using the `NSAffineTransform` class or Quartz 2D routines.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- isRotatedOrScaledFromBase` (page 3184)

Related Sample Code
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
`NSView.h`

compositingFilter

Returns the CoreImage filter that is used to composite the receiver's contents with the background

```
- (CIFilter *)compositingFilter
```

Return Value
The CoreImage filter.

Discussion
This method returns the value of the `filters` property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

contentFilters

Returns the array of CoreImage filters that are applied to the contents of the receiver and its sublayers.

```
- (NSArray *)contentFilters
```

Return Value
An array of CoreImage filters

Discussion
This method returns the value of the `filters` property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSView.h

convertPoint:fromView:

Converts a point from the coordinate system of a given view to that of the receiver.

```
- (NSPoint)convertPoint:(NSPoint)aPoint
    fromView:(NSView *)aView
```

Parameters

aPoint

A point specifying a location in the coordinate system of aView.

aView

The view with aPoint in its coordinate system. If *aView* is nil, this method instead converts from window base coordinates. Otherwise, both *aView* and the receiver must belong to the same NSWindow object.

Return Value
The point converted to the coordinate system of the receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- convertRect:fromView: (page 3157)
- convertSize:fromView: (page 3159)
- ancestorSharedWithView: (page 3147)

contentView (page 3313) (NSWindow)

Related Sample Code
CIAnnotation
CircleView
Cocoa OpenGL
GLUT
Sketch-112

Declared In
NSView.h

convertPoint:toView:

Converts a point from the receiver's coordinate system to that of a given view.

```
- (NSPoint)convertPoint:(NSPoint)aPoint
    toView:(NSView *)aView
```

Parameters

aPoint

A point specifying a location in the coordinate system of the receiver.

aView

The view into whose coordinate system *aPoint* is to be converted. If *aView* is `nil`, this method instead converts to window base coordinates. Otherwise, both *aView* and the receiver must belong to the same `NSWindow` object.

Return Value
The point converted to the coordinate system of *aView*.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- convertRect:toView:` (page 3158)
`- convertSize:toView:` (page 3160)
`- ancestorSharedWithView:` (page 3147)
`contentView` (page 3313) (`NSWindow`)

Related Sample Code
GLUT

Declared In
`NSView.h`

convertPointFromBase:

Converts the point from the base coordinate system to the receiver's coordinate system.

```
- (NSPoint)convertPointFromBase:(NSPoint)aPoint
```

Parameters

aPoint

A point specifying a location in the base coordinate system.

Return Value
The point converted to the receiver's base coordinate system.

Discussion
See "The View Coordinate System" in *View Programming Guide* for a description of view coordinate to base coordinate conversion.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
Sketch+Accessibility

Declared In
`NSView.h`

convertPointToBase:

Converts the point from the receiver's coordinate system to the base coordinate system.

```
- (NSPoint)convertPointToBase:(NSPoint)aPoint
```

Parameters

aPoint

A point specifying a location in the coordinate system of the receiver.

Return Value

The point converted to the base coordinate system.

Discussion

See "The View Coordinate System" in *View Programming Guide* for a description of view coordinate to base coordinate conversion.

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

Sketch+Accessibility

ZipBrowser

Declared In

NSView.h

convertRect:fromView:

Converts a rectangle from the coordinate system of another view to that of the receiver.

```
- (NSRect)convertRect:(NSRect)aRect
    fromView:(NSView *)aView
```

Parameters

aRect

The rectangle in `aView`'s coordinate system.

aView

The view with `aRect` in its coordinate system. If *aView* is `nil`, this method instead converts from window base coordinates. Otherwise, both *aView* and the receiver must belong to the same `NSWindow` object.

Return Value

The converted rectangle.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `convertPoint:fromView:` (page 3155)
- `convertSize:fromView:` (page 3159)
- `ancestorSharedWithView:` (page 3147)

`contentView` (page 3313) (`NSWindow`)

Related Sample Code
VideoViewer

Declared In
`NSView.h`

convertRect:toView:

Converts a rectangle from the receiver's coordinate system to that of another view.

```
- (NSRect)convertRect:(NSRect)aRect
    toView:(NSView *)aView
```

Parameters

aRect

A rectangle in the receiver's coordinate system.

aView

The view that is the target of the conversion operation. If *aView* is `nil`, this method instead converts from window base coordinates. Otherwise, both *aView* and the receiver must belong to the same `NSWindow` object.

Return Value
The converted rectangle.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `convertPoint:toView:` (page 3155)
- `convertSize:toView:` (page 3160)
- `ancestorSharedWithView:` (page 3147)

`contentView` (page 3313) (`NSWindow`)

Related Sample Code
CocoaAUHost
GLUT
Rulers
Sketch-112
VideoViewer

Declared In
`NSView.h`

convertRectFromBase:

Converts the rectangle from the base coordinate system to the receiver's coordinate system.

```
- (NSRect)convertRectFromBase:(NSRect)aRect
```

Parameters

aRect

A rectangle in the base coordinate system

Return Value

A rectangle in the receiver's coordinate system

Discussion

See "The View Coordinate System" in *View Programming Guide* for a description of view coordinate to base coordinate conversion.

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

Cocoa Tips and Tricks

Declared In

`NSView.h`

convertRectToBase:

Converts the rectangle from the receiver's coordinate system to the base coordinate system.

```
- (NSRect)convertRectToBase:(NSRect)aRect
```

Parameters

aRect

A rectangle in the receiver's coordinate system

Return Value

A rectangle in the base coordinate system

Discussion

See "The View Coordinate System" in *View Programming Guide* for a description of view coordinate to base coordinate conversion.

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

AnimatedTableView

Cocoa Tips and Tricks

Declared In

`NSView.h`

convertSize:fromView:

Converts a size from another view's coordinate system to that of the receiver.

```
- (NSSize)convertSize:(NSSize)aSize
    fromView:(NSView *)aView
```

Parameters

aSize

The size (width and height) in *aView*'s coordinate system.

aView

The view with *aSize* in its coordinate system. If *aView* is `nil`, this method instead converts from window base coordinates. Otherwise, both *aView* and the receiver must belong to the same `NSWindow` object.

Return Value
The converted size, as an `NSSize` structure.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- convertPoint:fromView:` (page 3155)
`- convertRect:fromView:` (page 3157)
`- ancestorSharedWithView:` (page 3147)
`contentView` (page 3313) (`NSWindow`)

Declared In
`NSView.h`

convertSize:toView:

Converts a size from the receiver's coordinate system to that of another view.

```
- (NSSize)convertSize:(NSSize)aSize
    toView:(NSView *)aView
```

Parameters

aSize

The size (width and height) in the receiver's coordinate system.

aView

The view that is the target of the conversion operation. If *aView* is `nil`, this method instead converts from window base coordinates. Otherwise, both *aView* and the receiver must belong to the same `NSWindow` object.

Return Value
The converted size, as an `NSSize` structure.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- convertPoint:toView:` (page 3155)
`- convertRect:toView:` (page 3158)
`- ancestorSharedWithView:` (page 3147)
`contentView` (page 3313) (NSWindow)

Related Sample Code
PhotoSearch

PredicateEditorSample

Declared In
NSView.h

convertSizeFromBase:

Converts the size from the base coordinate system to the receiver's coordinate system.

```
- (NSSize)convertSizeFromBase:(NSSize)aSize
```

Parameters

aSize

A size in the base coordinate system

Return Value
The size converted to the receiver's coordinate system.

Discussion
See "The View Coordinate System" in *View Programming Guide* for a description of view coordinate to base coordinate conversion.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
Sketch+Accessibility

Declared In
NSView.h

convertSizeToBase:

Converts the size from the receiver's coordinate system to the base coordinate system.

```
- (NSSize)convertSizeToBase:(NSSize)aSize
```

Parameters

aSize

A size in the receiver's coordinate system

Return Value
The size converted to the base coordinate system.

Discussion
See "The View Coordinate System" in *View Programming Guide* for a description of view coordinate to base coordinate conversion.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
Sketch+Accessibility
ZipBrowser

Declared In
NSView.h

dataWithEPSInsideRect:

Returns EPS data that draws the region of the receiver within a specified rectangle.

```
- (NSData *)dataWithEPSInsideRect:(NSRect)aRect
```

Parameters

aRect

A rectangle defining the region.

Discussion
This data can be placed on an NSPasteboard object, written to a file, or used to create an NSImage object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- writeEPSInsideRect:toPasteboard: (page 3248)

Declared In
NSView.h

dataWithPDFInsideRect:

Returns PDF data that draws the region of the receiver within a specified rectangle.

```
- (NSData *)dataWithPDFInsideRect:(NSRect)aRect
```

Parameters

aRect

A rectangle defining the region.

Discussion
This data can be placed on an NSPasteboard object, written to a file, or used to create an NSImage object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- writePDFInsideRect:toPasteboard: (page 3249)

Related Sample Code
QuickLookSketch
Sketch+Accessibility
Sketch-112

Declared In
NSView.h

didAddSubview:

Overridden by subclasses to perform additional actions when subviews are added to the receiver.

```
- (void)didAddSubview:(NSView *)subview
```

Parameters

subview

The view that was added as a subview.

Discussion

This method is invoked by `addSubview:` (page 3139).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSView.h`

discardCursorRects

Invalidates all cursor rectangles set up using `addCursorRect:cursor:` (page 3139).

```
- (void)discardCursorRects
```

Discussion

You need never invoke this method directly; neither is it typically invoked during the invalidation of cursor rectangles. `NSWindow` automatically invalidates cursor rectangles in response to `invalidateCursorRectsForView:` (page 3335) and before the receiver's cursor rectangles are reestablished using `resetCursorRects` (page 3206). This method is invoked just before the receiver is removed from a window and when the receiver is deallocated.

Availability

Available in Mac OS X v10.0 and later.

See Also

`discardCursorRects` (page 3320) (NSWindow)

Related Sample Code

DragItemAround

Declared In

`NSView.h`

display

Displays the receiver and all its subviews if possible, invoking each the `NSView` methods `lockFocus` (page 3187), `drawRect:` (page 3170), and `unlockFocus` (page 3239) as necessary.

```
- (void)display
```

Discussion

If the receiver isn't opaque, this method backs up the view hierarchy to the first opaque ancestor, calculates the portion of the opaque ancestor covered by the receiver, and begins displaying from there.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `canDraw` (page 3152)
- `opaqueAncestor` (page 3193)
- `visibleRect` (page 3245)
- `displayIfNeededIgnoringOpacity` (page 3165)

Related Sample Code
CIVideoDemoGL
CocoaSpeechSynthesisExample
EnhancedAudioBurn
GLUT
QTGraphicsImport

Declared In
`NSView.h`

displayIfNeeded

Displays the receiver and all its subviews if any part of the receiver has been marked as needing display with a `setNeedsDisplay:` (page 3225) or `setNeedsDisplayInRect:` (page 3225) message.

```
- (void)displayIfNeeded
```

Discussion
This method invokes the `NSView` methods `lockFocus` (page 3187), `drawRect:` (page 3170), and `unlockFocus` (page 3239) as necessary. If the receiver isn't opaque, this method backs up the view hierarchy to the first opaque ancestor, calculates the portion of the opaque ancestor covered by the receiver, and begins displaying from there.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `display` (page 3163)
- `needsDisplay` (page 3191)
- `displayIfNeededIgnoringOpacity` (page 3165)

Related Sample Code
ButtonMadness
CompositeLab
MixMash
Rulers

Declared In
`NSView.h`

displayIfNeededIgnoringOpacity

Acts as `displayIfNeeded` (page 3164), except that this method doesn't back up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.

```
- (void)displayIfNeededIgnoringOpacity
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DispatchFractal

Declared In
`NSView.h`

displayIfNeededInRect:

Acts as `displayIfNeeded` (page 3164), confining drawing to a specified region of the receiver..

```
- (void)displayIfNeededInRect:(NSRect)aRect
```

Parameters

aRect

A rectangle defining the region to be redrawn. It should be specified in the coordinate system of the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

displayIfNeededInRectIgnoringOpacity:

Acts as `displayIfNeeded` (page 3164), but confining drawing to *aRect* and not backing up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.

```
- (void)displayIfNeededInRectIgnoringOpacity:(NSRect)aRect
```

Parameters

aRect

A rectangle defining the region to be redrawn. It should be specified in the coordinate system of the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

displayRect:

Acts as `display` (page 3163), but confining drawing to a rectangular region of the receiver.

```
- (void)displayRect:(NSRect)aRect
```

Parameters

aRect

A rectangle defining the region of the receiver to be redrawn; should be specified in the coordinate system of the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSView.h

displayRectIgnoringOpacity:

Displays the receiver but confines drawing to a specified region and does not back up to the first opaque ancestor—it simply causes the receiver and its descendants to execute their drawing code.

```
- (void)displayRectIgnoringOpacity:(NSRect)aRect
```

Parameters

aRect

A rectangle defining the region of the receiver to be redrawn; should be specified in the coordinate system of the receiver.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSView.h

displayRectIgnoringOpacity:inContext:

Causes the receiver and its descendants to be redrawn to the specified graphics context.

```
- (void)displayRectIgnoringOpacity:(NSRect)aRect
    inContext:(NSGraphicsContext *)context
```

Parameters

aRect

A rectangle defining the region of the receiver to be redrawn. It should be specified in the coordinate system of the receiver.

context

The graphics context in which drawing will occur. See the discussion below for more about this parameter.

Discussion
Acts as `display` (page 3163), but confines drawing to *aRect*. This method initiates drawing with the receiver, even if the receiver is not opaque. Appropriate scaling factors for the view are obtained from *context*.

If the *context* parameter represents the context for the window containing the view, then all of the necessary transformations are applied. This includes the application of the receiver's bounds and frame transforms along with any transforms it inherited from its ancestors. In this situation, the view is also marked as no longer needing an update for the specified rectangle.

If *context* specifies any other graphics context, then only the receiver's bounds transform is applied. This means that drawing is not constrained to the view's visible rectangle. It also means that any dirty rectangles are not cleared, since they are not being redrawn to the window.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
QuickLookSketch

Declared In
NSView.h

dragFile:fromRect:slideBack:event:

Initiates a dragging operation from the receiver, allowing the user to drag a file icon to any application that has window or view objects that accept files.

```
- (BOOL)dragFile:(NSString *)fullPath fromRect:(NSRect)aRect
    slideBack:(BOOL)slideBack event:(NSEvent *)theEvent
```

Parameters

fullPath
A string that specifies the absolute path for the file that is dragged.

aRect
A rectangle that describes the position of the icon in the receiver's coordinate system.

slideBack
A Boolean that indicates whether the icon being dragged should slide back to its position in the receiver if the file isn't accepted. The icon slides back to *aRect* if *slideBack* is YES, the file is not accepted by the dragging destination, and the user has not disabled icon animation; otherwise it simply disappears.

theEvent
The mouse-down event object from which to initiate the drag operation. In particular, its mouse location is used for the offset of the icon being dragged.

Return Value
YES if the receiver successfully initiates the dragging operation (which doesn't necessarily mean the dragging operation concluded successfully). Otherwise returns NO.

Discussion
This method must be invoked only within an implementation of the mouseDown: (page 2164) method.

See the NSDraggingSource, NSDraggingInfo, and NSDraggingDestination protocol specifications for more information on dragging operations.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `dragImage:at:offset:event:pasteboard:source:slideBack:` (page 3168)
- `shouldDelayWindowOrderingForEvent:` (page 3231)

Declared In
`NSView.h`

dragImage:at:offset:event:pasteboard:source:slideBack:

Initiates a dragging operation from the receiver, allowing the user to drag arbitrary data with a specified icon into any application that has window or view objects that accept dragged data.

```
- (void)dragImage:(NSImage *)anImage at:(NSPoint)imageLoc offset:(NSSize)mouseOffset
    event:(NSEvent *)theEvent pasteboard:(NSPasteboard *)pboard
   source:(id)sourceObject slideBack:(BOOL)slideBack
```

Parameters

anImage
The `NSImage` object to be dragged.

imageLoc
The location of the image's lower-left corner, in the receiver's coordinate system. It determines the placement of the dragged image under the cursor. When determining the image location you should use the mouse down coordinate, provided in *theEvent*, rather than the current mouse location.

mouseOffset
This parameter is ignored.

theEvent
The left mouse-down event that triggered the dragging operation (see discussion below).

pboard
The pasteboard that holds the data to be transferred to the destination (see discussion below).

sourceObject
An object that serves as the controller of the dragging operation. It must conform to the `NSDraggingSource` informal protocol and is typically the receiver itself or its `NSWindow` object.

slideBack
A Boolean that determines whether the drag image should slide back if it's rejected. The image slides back to *imageLoc* if *slideBack* is `YES` and the image isn't accepted by the dragging destination. If `NO` the image doesn't slide back.

Discussion
This method must be invoked only within an implementation of the `mouseDown:` (page 2164) or `mouseDragged:` (page 2164) methods.

Before invoking this method, you must place the data to be transferred on *pboard*. To do this, get the drag pasteboard object (`NSDragPboard`), declare the types of the data, and then put the data on the pasteboard. This code fragment initiates a dragging operation on an image itself (that is, the image is the data to be transferred):

```
- (void)mouseDown:(NSEvent *)theEvent
{
    NSSize dragOffset = NSMakeSize(0.0, 0.0);
    NSPasteboard *pboard;
```

```
    pboard = [NSPasteboard pasteboardWithName:NSDragPboard];
    [pboard declareTypes:[NSArray arrayWithObject:NSTIFFPboardType]  owner:self];
    [pboard setData:[[self image] TIFFRepresentation] forType:NSTIFFPboardType];

    [self dragImage:[self image] at:[self imageLocation] offset:dragOffset
        event:theEvent pasteboard:pboard source:self slideBack:YES];

    return;
}
```

See the `NSDraggingSource`, `NSDraggingInfo`, and `NSDraggingDestination` protocol specifications for more information on dragging operations.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `dragFile:fromRect:slideBack:event:` (page 3167)
- `shouldDelayWindowOrderingForEvent:` (page 3231)

Declared In
`NSView.h`

dragPromisedFilesOfTypes:fromRect:source:slideBack:event:

Initiates a dragging operation from the receiver, allowing the user to drag one or more promised files (or directories) into any application that has window or view objects that accept promised file data.

```
- (BOOL)dragPromisedFilesOfTypes:(NSArray *)typeArray fromRect:(NSRect)aRect
    source:(id)sourceObject slideBack:(BOOL)slideBack event:(NSEvent *)theEvent
```

Parameters

typeArray
An array of file types being promised. The array elements can consist of file extensions and HFS types encoded with the `NSFileTypeForHFSTypeCode` function. If promising a directory of files, only include the top directory in the array.

aRect
A rectangle that describes the position of the icon in the receiver's coordinate system.

sourceObject
An object that serves as the controller of the dragging operation. It must conform to the `NSDraggingSource` informal protocol, and is typically the receiver itself or its `NSWindow` object.

slideBack
A Boolean that indicates whether the icon being dragged should slide back to its position in the receiver if the file isn't accepted. The icon slides back to *aRect* if *slideBack* is `YES`, the promised files are not accepted by the dragging destination, and the user has not disabled icon animation; otherwise it simply disappears.

theEvent
The mouse-down event object from which to initiate the drag operation. In particular, its mouse location is used for the offset of the icon being dragged.

Return Value
`YES` if the drag operation is initiated successfully, `NO` otherwise.

Discussion
This method must be invoked only within an implementation of the `mouseDown:` (page 2164) method. As part of its implementation, this method invokes `dragImage:at:offset:event:pasteboard:source:slideBack:` (page 3168).

Promised files are files that do not exist, yet, but that the drag source, *sourceObject*, promises to create at a file system location specified by the drag destination when the drag is successfully dropped.

See *Drag and Drop Programming Topics for Cocoa* for more information on dragging operations.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `dragImage:at:offset:event:pasteboard:source:slideBack:` (page 3168)
- `shouldDelayWindowOrderingForEvent:` (page 3231)

Declared In
`NSView.h`

drawPageBorderWithSize:

Allows applications that use the Application Kit pagination facility to draw additional marks on each logical page.

```
- (void)drawPageBorderWithSize:(NSSize)borderSize
```

Parameters
borderSize
An `NSSize` structure that defines a logical page.

Discussion
The marks can be such things as alignment marks or a virtual sheet border of size *borderSize*. The default implementation doesn't draw anything.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawSheetBorderWithSize:` (page 3171)

Declared In
`NSView.h`

drawRect:

Overridden by subclasses to draw the receiver's image within the passed-in rectangle.

```
- (void)drawRect:(NSRect)dirtyRect
```

Parameters
dirtyRect
A rectangle defining the dirty area of the view that requires redrawing.

Discussion
The receiver can assume the focus has been locked and the coordinate transformations of its frame and bounds rectangles have been applied; all it needs to do is invoke rendering client functions.

You can avoid unnecessary drawing by paying attention to the *dirtyRect* parameter which defines the area that must be redrawn. The bounds of the entire view, as opposed to the dirty region, is given by [self bounds].

On Mac OS X version 10.2 and earlier, the Application Kit automatically clips any drawing you perform in this method to this rectangle. On Mac OS X version 10.3 and later, the Application Kit automatically clips drawing to a list of non-overlapping rectangles that more rigorously specify the area needing drawing. You can invoke the `getRectsBeingDrawn:count:` (page 3176) method to retrieve this list of rectangles and use them to constrain your drawing more tightly, if you wish. Moreover, the `needsToDrawRect:` (page 3192) method gives you a convenient way to test individual objects for intersection with the rectangles in the list. See *Cocoa Drawing Guide* for information and references on drawing.

The default implementation does nothing. If your custom view is a direct `NSView` subclass you do not need to call `super`'s implementation.

Note: If a subclass returns `YES` upon receiving an `isOpaque` (page 3183) message, it must completely fill *dirtyRect* with opaque content.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `display` (page 3163)
- `getRectsBeingDrawn:count:` (page 3176)
- `isFlipped` (page 3181)
- `needsToDrawRect:` (page 3192)
- `setNeedsDisplayInRect:` (page 3225)
- `shouldDrawColor` (page 3232)

Related Sample Code
QTCoreImage101
QTCoreVideo101
QTCoreVideo103
QTCoreVideo202
Transformed Image

Declared In
`NSView.h`

drawSheetBorderWithSize:

Allows applications that use the Application Kit pagination facility to draw additional marks on each printed sheet.

```
- (void)drawSheetBorderWithSize:(NSSize)borderSize
```

Parameters

borderSize

An `NSSize` structure that defines a printed sheet.

Discussion

The marks can be such things as crop marks or fold lines of size *borderSize*. This method has been deprecated.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- drawPageBorderWithSize:` (page 3170)

Declared In

`NSView.h`

enclosingMenuItem

Returns the menu item containing the receiver or any of its superviews in the view hierarchy.

```
- (NSMenuItem *)enclosingMenuItem
```

Return Value

Returns the menu item containing the receiver or any of its superviews in the view hierarchy, or `nil` if the receiver's view hierarchy is not in a menu item

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSMenuItem.h`

enclosingScrollView

Returns the nearest ancestor `NSScrollView` object containing the receiver (not including the receiver itself); otherwise returns `nil`.

```
- (NSScrollView *)enclosingScrollView
```

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

PhotoSearch

PredicateEditorSample

Rulers

Sketch+Accessibility

Sketch-112

Declared In

`NSView.h`

endDocument

This method is invoked at the end of the printing session.

```
- (void)endDocument
```

Discussion
If you override this method, call the superclass implementation.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

endPage

Writes the end of a conforming page.

```
- (void)endPage
```

Discussion
This method is invoked after each page is printed. It invokes `unlockFocus` (page 3239). This method also generates comments for the bounding box and page fonts, if they were specified as being at the end of the page.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

enterFullScreenMode:withOptions:

Sets the receiver to full screen mode.

```
- (BOOL)enterFullScreenMode:(NSScreen *)screen
    withOptions:(NSDictionary *)options
```

Parameters

screen
The screen the receiver should cover.

options
A dictionary of options for the mode. For possible keys, see "`Full Screen Mode Options`" (page 3252).

Return Value
`YES` if the receiver was able to enter full screen mode, otherwise `NO`.

Discussion
When the `NSFullScreenModeApplicationPresentationOptions` (page 3252) is contained in the options dictionary, the presentation options that were in effect when this method is invoked are not altered, and no displays are captured.

If you do not wish to capture the screen when going to full screen mode, you can add `NSFullScreenModeApplicationPresentationOptions` (page 3252) to the options dictionary with the value returned by the `presentationOptions` (page 158).

When the `NSFullScreenModeApplicationPresentationOptions` (page 3252) options is specified, exiting full screen mode using `exitFullScreenModeWithOptions:` (page 3174) will restore the previously active `presentationOptions` (page 158).

Special Considerations

On Mac OS X v 10.5 invoking this method when the receiver was not in a window would cause an exception. On Mac OS X v 10.6 and later, you can now send this message to a view not in a window. For applications that must also run on Mac OS X v 10.5, a simple workaround is to place the view in an offscreen dummy window.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `exitFullScreenModeWithOptions:` (page 3174)
- `isInFullScreenMode` (page 3183)

Related Sample Code
CoreAnimationKioskStyleMenu
From A View to A Movie
From A View to A Picture
GeekGameBoard

Declared In
`NSView.h`

exitFullScreenModeWithOptions:

Instructs the receiver to exit full screen mode.

```
- (void)exitFullScreenModeWithOptions:(NSDictionary *)options
```

Parameters

options

A dictionary of options for the mode. For possible keys, see "`Full Screen Mode Options`" (page 3252).

Discussion
When the `NSFullScreenModeApplicationPresentationOptions` (page 3252) options is specified when `enterFullScreenMode:withOptions:` (page 3173) is invoked, exiting full screen mode will restore the previously active `presentationOptions` (page 158).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `enterFullScreenMode:withOptions:` (page 3173)
- `isInFullScreenMode` (page 3183)

Related Sample Code
From A View to A Movie
From A View to A Picture
GeekGameBoard

Declared In
`NSView.h`

focusRingType

Returns the type of focus ring drawn around the receiver.

```
- (NSFocusRingType)focusRingType
```

Return Value
An `enum` constant identifying a type of focus ring. Possible values are listed in `NSFocusRingType` (page 4009).

Discussion
You can disable a view's drawing of its focus ring by overriding this method to return `NSFocusRingTypeNone`, or by invoking `setFocusRingType:` (page 3218) with and argument of `NSFocusRingTypeNone`. You should only disable the default drawing of a view's focus ring if you want it to draw its own focus ring (for example, setting the background color of the view), or if the view does not have sufficient space to display a focus ring.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setFocusRingType:` (page 3218)

Declared In
`NSView.h`

frame

Returns the receiver's frame rectangle, which defines its position in its superview.

```
- (NSRect)frame
```

Discussion
The frame rectangle may be rotated; use the `frameRotation` (page 3176) method to check this.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- bounds` (page 3150)
`- setFrame:` (page 3218)

Related Sample Code
FunHouse
MyPhoto

Quartz Composer QCTV
Quartz Composer WWDC 2005 TextEdit
TextSizingExample

Declared In
`NSView.h`

frameCenterRotation

Returns the receiver's rotation about the layer's position.

```
- (CGFloat)frameCenterRotation
```

Return Value
The angle of rotation of the frame around the center of the receiver.

Discussion
If the application has altered the layer's `anchorPoint` property, the behavior is undefined. Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

frameRotation

Returns the angle, in degrees, of the receiver's frame relative to its superview's coordinate system.

```
- (CGFloat)frameRotation
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFrameRotation:` (page 3221)
- `boundsRotation` (page 3151)

Declared In
`NSView.h`

getRectsBeingDrawn:count:

Returns by indirection a list of non-overlapping rectangles that define the area the receiver is being asked to draw in `drawRect:` (page 3170).

```
- (void)getRectsBeingDrawn:(const NSRect **)rects
    count:(NSInteger *)count
```

Parameters

rects

On return, contains a list of non-overlapping rectangles defining areas to be drawn in. The rectangles returned in *rects* are in the coordinate space of the receiver.

count

On return, the number of rectangles in the `rects` list.

Discussion

An implementation of `drawRect:` can use this information to test whether objects or regions within the view intersect with the rectangles in the list, and thereby avoid unnecessary drawing that would be completely clipped away.

The `needsToDrawRect:` (page 3192) method gives you a convenient way to test individual objects for intersection with the area being drawn in `drawRect:` (page 3170). However, you may want to retrieve and directly inspect the rectangle list if this is a more efficient way to perform intersection testing.

You should send this message only from within a `drawRect:` (page 3170) implementation. The *aRect* parameter of `drawRect:` is the rectangle enclosing the returned list of rectangles; you can use it in an initial pass to reject objects that are clearly outside the area to be drawn.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- wantsDefaultClipping` (page 3246)

Declared In

`NSView.h`

getRectsExposedDuringLiveResize:count:

Returns a list of rectangles indicating the newly exposed areas of the receiver.

```
- (void)getRectsExposedDuringLiveResize:(NSRect)exposedRects
    count:(NSInteger *)count
```

Parameters

exposedRects

On return, contains the list of rectangles. The returned rectangles are in the coordinate space of the receiver.

count

Contains the number of rectangles in *exposedRects*; this value may be 0 and is guaranteed to be no more than 4.

Discussion

If your view does not support content preservation during live resizing, the entire area of your view is returned in the *exposedRects* parameter. To support content preservation, override `preservesContentDuringLiveResize` (page 3196) in your view and have your implementation return `YES`.

Note: The window containing your view must also support content preservation. To enable support for this feature in your window, use the `setPreservesContentDuringLiveResize:` (page 3393) method of `NSWindow`.

If the view decreased in both height and width, the list of returned rectangles will be empty. If the view increased in both height and width and its upper-left corner stayed anchored in the same position, the list of returned rectangles will contain a vertical and horizontal component indicating the exposed area.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `preservesContentDuringLiveResize` (page 3196)
- `rectPreservedDuringLiveResize` (page 3199)

Declared In
`NSView.h`

gState

Returns the identifier for the receiver's graphics state object, or 0 if the receiver doesn't have a graphics state object.

```
- (NSInteger)gState
```

Discussion
A view object's graphics state object is recreated from scratch whenever the view is focused, unless the `allocateGState` (page 3146) method has been invoked. So if the receiver hasn't been focused or hasn't received the `allocateGState` (page 3146) message, this method returns 0.

Although applications rarely need to use the value returned by `gState` (page 3178), it can be passed to the few methods that take an object identifier as a parameter.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `allocateGState` (page 3146)
- `setUpGState` (page 3229)
- `renewGState` (page 3205)
- `releaseGState` (page 3201)
- `lockFocus` (page 3187)

Declared In
`NSView.h`

heightAdjustLimit

Returns the fraction (from 0.0 to 1.0) of the page that can be pushed onto the next page during automatic pagination to prevent items such as lines of text from being divided across pages.

```
- (CGFloat)heightAdjustLimit
```

Discussion
This fraction is used to calculate the bottom edge limit for an `adjustPageHeightNew:top:bottom:limit:` (page 3143) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- widthAdjustLimit` (page 3247)

Declared In
`NSView.h`

hitTest:

Returns the farthest descendant of the receiver in the view hierarchy (including itself) that contains a specified point, or `nil` if that point lies completely outside the receiver.

```
- (NSView *)hitTest:(NSPoint)aPoint
```

Parameters

aPoint

A point that is in the coordinate system of the receiver's superview, not of the receiver itself.

Return Value
A view object that is the farthest descendent of *aPoint*.

Discussion
This method is used primarily by an `NSWindow` object to determine which view should receive a mouse-down event. You'd rarely need to invoke this method, but you might want to override it to have a view object hide mouse-down events from its subviews. This method ignores hidden views.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- mouse:inRect:` (page 3189)
`- convertPoint:toView:` (page 3155)
`- setHidden:` (page 3222)

Related Sample Code
TargetGallery

Declared In
`NSView.h`

initWithFrame:

Initializes and returns a newly allocated `NSView` object with a specified frame rectangle.

```
- (id)initWithFrame:(NSRect)frameRect
```

Parameters

frameRect

The frame rectangle for the created view object.

Return Value

An initialized `NSView` object or nil if the object couldn't be created.

Discussion

The new view object must be inserted into the view hierarchy of a window before it can be used. This method is the designated initializer for the `NSView` class. Returns an initialized object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `addSubview:` (page 3139)
- `addSubview:positioned:relativeTo:` (page 3140)
- `setFrame:` (page 3218)

Related Sample Code

CocoaSlides

DotViewUndo

DragItemAround

Movie Overlay

Rulers

Declared In

`NSView.h`

inLiveResize

A convenience method, expected to be called from `drawRect:` (page 3170), to assist in decisions about optimized drawing.

```
- (BOOL)inLiveResize
```

Return Value

`YES` if the receiver is in a live-resize operation, `NO` otherwise.

Availability

Available in Mac OS X v10.1 and later.

See Also

- `viewDidEndLiveResize` (page 3240)
- `viewWillStartLiveResize` (page 3244)

Related Sample Code

Cocoa OpenGL

MatrixMixerTest

WhackedTV

Declared In

`NSView.h`

inputContext

Returns the text input context object for the receiver.

```
- (NSTextInputContext *)inputContext
```

Return Value
The text input context object, or `nil` the receiver doesn't conform to `NSTextInputClient` protocol.

Availability
Available in Mac OS X v10.6 and later.

Related Sample Code
TextInputView

Declared In
`NSView.h`

isDescendantOf:

Returns `YES` if the receiver is a subview of a given view or if it's identical to that view; otherwise, it returns `NO`.

```
- (BOOL)isDescendantOf:(NSView *)aView
```

Parameters

aView
The view to test for subview relationship within the view hierarchy.

Discussion
The method returns `YES` if the receiver is either an immediate or distant subview of `aView`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `superview` (page 3235)
- `subviews` (page 3235)
- `ancestorSharedWithView:` (page 3147)

Declared In
`NSView.h`

isFlipped

Returns `YES` if the receiver uses flipped drawing coordinates or `NO` if it uses native coordinates.

```
- (BOOL)isFlipped
```

Discussion
The default implementation returns `NO`; subclasses that use flipped coordinates should override this method to return `YES`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ImageMap
ImageMapExample
Quartz Composer WWDC 2005 TextEdit
Rulers
Sketch-112

Declared In
`NSView.h`

isHidden

Returns whether the receiver is marked as hidden.

```
- (BOOL)isHidden
```

Discussion
The return value reflects the state of the receiver only, as set in Interface Builder or through the most recent `setHidden:` (page 3222) message, and does not account for the state of the receiver's ancestors in the view hierarchy, Thus this method returns `NO` when the receiver is effectively hidden because it has a hidden ancestor. See `setHidden:` for a discussion of the mechanics and implications of hidden views.

If you want to determine whether a view is effectively hidden, for whatever reason, send the `isHiddenOrHasHiddenAncestor` (page 3182) to the view instead.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
GLSLShowpiece

Declared In
`NSView.h`

isHiddenOrHasHiddenAncestor

Returns `YES` if the receiver is marked as hidden or has an ancestor in the view hierarchy that is marked as hidden; returns `NO` otherwise.

```
- (BOOL)isHiddenOrHasHiddenAncestor
```

Discussion
The return value reflects state set through the `setHidden:` (page 3222) method in the receiver of one of its ancestors in the view hierarchy. It does not account for other reasons why a view might be considered hidden, such as being positioned outside its superview's bounds, not having a window, or residing in a window that is offscreen or overlapped by another window.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `isHidden` (page 3182)

Declared In
`NSView.h`

isInFullScreenMode

Returns whether the view is in full screen mode.

```
- (BOOL)isInFullScreenMode
```

Return Value
`YES` if the receiver is in full screen mode, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `enterFullScreenMode:withOptions:` (page 3173)
- `exitFullScreenModeWithOptions:` (page 3174)

Related Sample Code
Denoise
From A View to A Movie
GeekGameBoard
OpenCL NBody Simulation Example
QTCoreVideo201

Declared In
`NSView.h`

isOpaque

Overridden by subclasses to return `YES` if the receiver is opaque, `NO` otherwise.

```
- (BOOL)isOpaque
```

Discussion
A view object is opaque if it completely covers its frame rectangle when drawing itself. The default implementation performs no drawing at all and so returns `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `opaqueAncestor` (page 3193)
- `displayRectIgnoringOpacity:` (page 3166)
- `displayIfNeededIgnoringOpacity` (page 3165)
- `displayIfNeededInRectIgnoringOpacity:` (page 3165)

Related Sample Code
ImageApp
Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility
Sketch-112
WhackedTV

Declared In
`NSView.h`

isRotatedFromBase

Returns `YES` if the receiver or any of its ancestors has ever received a `setFrameRotation:` (page 3221) or `setBoundsRotation:` (page 3215) message; otherwise returns `NO`.

```
- (BOOL)isRotatedFromBase
```

Discussion
The intent of this information is to optimize drawing and coordinate calculation, not necessarily to reflect the exact state of the receiver's coordinate system, so it may not reflect the actual rotation. For example, if an `NSView` object is rotated to 45 degrees and later back to 0, this method still returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `frameRotation` (page 3176)
- `boundsRotation` (page 3151)

Declared In
`NSView.h`

isRotatedOrScaledFromBase

Returns `YES` if the receiver or any of its ancestors has ever had a nonzero frame or bounds rotation, or has been scaled from the window's base coordinate system; otherwise returns `NO`.

```
- (BOOL)isRotatedOrScaledFromBase
```

Discussion
The intent of this information is to optimize drawing and coordinate calculation, not necessarily to reflect the exact state of the receiver's coordinate system, so it may not reflect the actual rotation or scaling. For example, if an `NSView` object is rotated to 45 degrees and later back to 0, this method still returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `frameRotation` (page 3176)
- `boundsRotation` (page 3151)
- `centerScanRect:` (page 3153)

- setBounds: (page 3214)
- setBoundsSize: (page 3216)
- scaleUnitSquareToSize: (page 3208)

Declared In
NSView.h

knowsPageRange:

Returns YES if the receiver handles page boundaries, NO otherwise.

```
- (BOOL)knowsPageRange:(NSRangePointer)aRange
```

Parameters

aRange

On return, holds the page range if YES is returned directly. Page numbers are one-based—that is pages run from one to *N*.

Discussion
Returns NO if the receiver uses the default auto-pagination mechanism. The default implementation returns NO. Override this method if your class handles page boundaries.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSView.h

layer

Returns the Core Animation layer that the receiver uses as its backing store.

```
- (CALayer *)layer
```

Return Value
The Core Animation layer the receiver is using as its backing store.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
AnimatedTableView
CALayerEssentials
FunHouse
GeekGameBoard
Quartz2DBasics

Declared In
NSView.h

layerContentsPlacement

Returns the current layer contents placement policy.

```
- (NSViewLayerContentsPlacement)layerContentsPlacement
```

Return Value
The placement policy. See "NSViewLayerContentsPlacement" (page 3253) for supported values.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setLayerContentsPlacement:` (page 3223)

Declared In
`NSView.h`

layerContentsRedrawPolicy

Returns the view's layer contents redraw policy.

```
- (NSViewLayerContentsRedrawPolicy)layerContentsRedrawPolicy
```

Return Value
The current redraw policy. See "NSViewLayerContentsRedrawPolicy" (page 3252) for supported values.

Discussion
The `layerContentsRedrawPolicy` (page 3186) and `layerContentsPlacement` (page 3186) settings can have significant impacts on performance. See `setLayerContentsRedrawPolicy:` (page 3224) and `setLayerContentsPlacement:` (page 3223) for more information.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setLayerContentsRedrawPolicy:` (page 3224)

Declared In
`NSView.h`

locationOfPrintRect:

Invoked by `print:` (page 3198) to determine the location of the region of the receiver being printed on the physical page.

```
- (NSPoint)locationOfPrintRect:(NSRect)aRect
```

Parameters

aRect

A rectangle defining a region of the receiver; it is expressed in the default coordinate system of the page.

Return Value
A point to be used for setting the origin for *aRect*, whose size the receiver can examine in order to properly place it. It is expressed in the default coordinate system of the page.

Discussion
The default implementation places *aRect* according to the status of the `NSPrintInfo` object for the print job. By default it places the image in the upper-left corner of the page, but if the `NSPrintInfo` methods `isHorizontallyCentered` (page 2051) or `isVerticallyCentered` (page 2052) return `YES`, it centers a single-page image along the appropriate axis. A multiple-page document, however, is always placed so the divided pieces can be assembled at their edges.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

lockFocus

Locks the focus on the receiver, so subsequent commands take effect in the receiver's window and coordinate system.

```
- (void)lockFocus
```

Discussion
If you don't use a `display...` method to draw an `NSView` object, you must invoke `lockFocus` before invoking methods that send commands to the window server, and must balance it with an `unlockFocus` (page 3239) message when finished.

Hiding or miniaturizing a one-shot window causes the backing store for that window to be released. If you don't use the standard display mechanism to draw, you should use `lockFocusIfCanDraw` (page 3188) rather than `lockFocus` if there is a chance of drawing while the window is either miniaturized or hidden.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ focusView` (page 3137)
`- display` (page 3163)
`- drawRect:` (page 3170)
`- lockFocusIfCanDraw` (page 3188)

Related Sample Code
CocoaAUHost
Color Sampler
GLUT
QTKitFrameStepper
VideoViewer

Declared In
`NSView.h`

lockFocusIfCanDraw

Locks the focus to the receiver atomically if the `canDraw` method returns `YES` and returns the value of `canDraw`.

```
- (BOOL)lockFocusIfCanDraw
```

Discussion
Your thread will not be preempted by other threads between the `canDraw` method and the lock. This method fails to lock focus and returns `NO`, when the receiver is hidden and the current context is drawing to the screen (as opposed to a printing context).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
GLUT
VideoViewer

Declared In
`NSView.h`

lockFocusIfCanDrawInContext:

Locks the focus to the receiver atomically if drawing can occur in the specified graphics context.

```
- (BOOL)lockFocusIfCanDrawInContext:(NSGraphicsContext *)context
```

Parameters
context
The graphics context in which drawing might occur. See the discussion for the implications of the type of context.

Return Value
`YES` if successful; otherwise, returns `NO`.

Discussion
Your thread will not be preempted by other threads between the `canDraw` method and the lock.

If the *context* parameter represents the context for the window containing the view, then all of the necessary transformations are applied. This includes the application of the receiver's bounds and frame transforms along with any transforms it inherited from its ancestors. If *context* specifies any other graphics context, then only the receiver's bounds transform is applied.

Special Considerations

Important: This method was declared in Mac OS X v10.4, but is not used in that release. It currently does nothing and returns `NO`. However, it might be implemented in a future release.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `lockFocus` (page 3187)

- lockFocusIfCanDraw (page 3188)

Declared In
NSView.h

makeBackingLayer

Creates the view's backing layer.

```
- (CALayer *)makeBackingLayer
```

Return Value
A layer to use as the view's backing layer.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSView.h

menuForEvent:

Overridden by subclasses to return a context-sensitive pop-up menu for a given mouse-down event.

```
- (NSMenu *)menuForEvent:(NSEvent *)theEvent
```

Parameters
theEvent
An object representing a mouse-down event.

Discussion
The receiver can use information in the mouse event, such as its location over a particular element of the receiver, to determine what kind of menu to return. For example, a text object might display a text-editing menu when the cursor lies over text and a menu for changing graphics attributes when the cursor lies over an embedded image.

The default implementation returns the receiver's normal menu.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ defaultMenu (page 3137)
menu (page 2163) (NSResponder)

Declared In
NSView.h

mouse:inRect:

Returns whether a region of the receiver contains a specified point, accounting for whether the receiver is flipped or not.

```
- (BOOL)mouse:(NSPoint)aPoint
    inRect:(NSRect)aRect
```

Parameters

aPoint

A point that is expressed in the receiver's coordinate system. This point generally represents the hot spot of the mouse cursor.

aRect

A rectangle that is expressed in the receiver's coordinate system.

Return Value

`YES` if `aRect` contains `aPoint`, `NO` otherwise.

Discussion

Point-in-rectangle functions generally assume that the bottom edge of a rectangle is outside of the rectangle boundaries, while the upper edge is inside the boundaries. This method views *aRect* from the point of view of the user—that is, this method always treats the bottom edge of the rectangle as the one closest to the bottom edge of the user's screen. By making this adjustment, this function ensures consistent mouse-detection behavior from the user's perspective.

Never use the Foundation's `NSPointInRect` function as a substitute for this method. It doesn't account for flipped coordinate systems.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- hitTest:` (page 3179)

`- isFlipped` (page 3181)

`NSMouseInRect` (Foundation functions)

`- convertPoint:fromView:` (page 3155)

Declared In

`NSView.h`

mouseDownCanMoveWindow

Returns `YES` if the receiver does not need to handle a mouse down and can pass it through to superviews; `NO` if it needs to handle the mouse down.

```
- (BOOL)mouseDownCanMoveWindow
```

Discussion

This allows iApp-type applications to determine the region by which a window can be moved. By default, this method returns `NO` if the view is opaque; otherwise, it returns `YES`. Subclasses can override this method to return a different value.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

DockTile

SpeedometerView

WebKitPluginStarter
WebKitPluginWithJavaScript

Declared In
NSView.h

needsDisplay

Returns YES if the receiver needs to be displayed, as indicated using setNeedsDisplay: (page 3225) and setNeedsDisplayInRect: (page 3225); returns NO otherwise.

```
- (BOOL)needsDisplay
```

Discussion
The displayIfNeeded... methods check this status to avoid unnecessary drawing, and all display methods clear this status to indicate that the view object is up to date.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSView.h

needsPanelToBecomeKey

Overridden by subclasses to determine if the receiver requires its panel, which might otherwise avoid becoming key, to become the key window so that it can handle keyboard input and navigation.

```
- (BOOL)needsPanelToBecomeKey
```

Return Value
Returns YES if it should be come key, NO otherwise.

Discussion
Such a subclass should also override acceptsFirstResponder (page 2143) to return YES.

This method is also used in keyboard navigation. It determines if a mouse click should give focus to a view (make it first responder). Some views will want to get keyboard focus when you click in them, for example text fields. Other views should only get focus if you tab to them, for example, buttons. You wouldn't want focus to shift from a textfield that has editing in progress simply because you clicked on a check box.

The default implementation returns NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
becomesKeyOnlyIfNeeded (page 1860) (NSPanel)

Related Sample Code
ClockControl

Declared In
NSView.h

needsToDrawRect:

Returns whether the specified rectangle intersects any part of the area that the receiver is being asked to draw.

```
- (BOOL)needsToDrawRect:(NSRect)aRect
```

Parameters

aRect

A rectangle defining a region of the receiver.

Discussion
You typically send this message from within a `drawRect:` (page 3170) implementation. It gives you a convenient way to determine whether any part of a given graphical entity might need to be drawn. It is optimized to efficiently reject any rectangle that lies outside the bounding box of the area the receiver is being asked to draw in `drawRect:`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSView.h

nextKeyView

Returns the view object following the receiver in the key view loop.

```
- (NSView *)nextKeyView
```

Return Value
Returns the view object following the receiver in the key view loop, or `nil` if there is none.

Discussion
This view should, if possible, be made first responder when the user navigates forward from the receiver using keyboard interface control.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- nextValidKeyView` (page 3193)
`- setNextKeyView:` (page 3226)
`- previousKeyView` (page 3197)
`- previousValidKeyView` (page 3197)
`selectNextKeyView:` (page 3364) (`NSWindow`)
`selectKeyViewFollowingView:` (page 3364) (`NSWindow`)
`selectPreviousKeyView:` (page 3365) (`NSWindow`)
`selectKeyViewPrecedingView:` (page 3364) (`NSWindow`)

Related Sample Code
TrackBall

Declared In
`NSView.h`

nextValidKeyView

Returns the closest view object in the key view loop that follows the receiver and accepts first responder status.

```
- (NSView *)nextValidKeyView
```

Return Value
The closest view object in the key view loop that follows the receiver and accepts first responder status, or `nil` if there is none.

Discussion
This method ignores hidden views when it determines the next valid key view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `nextKeyView` (page 3192)
- `setNextKeyView:` (page 3226)
- `previousKeyView` (page 3197)
- `previousValidKeyView` (page 3197)

`selectNextKeyView:` (page 3364) (`NSWindow`)
`selectKeyViewFollowingView:` (page 3364) (`NSWindow`)
`selectPreviousKeyView:` (page 3365) (`NSWindow`)
`selectKeyViewPrecedingView:` (page 3364) (`NSWindow`)
- `setHidden:` (page 3222)

Related Sample Code
Dicey

Declared In
`NSView.h`

opaqueAncestor

Returns the receiver's closest opaque ancestor (including the receiver itself).

```
- (NSView *)opaqueAncestor
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isOpaque` (page 3183)

- displayRectIgnoringOpacity: (page 3166)
- displayIfNeededIgnoringOpacity (page 3165)
- displayIfNeededInRectIgnoringOpacity: (page 3165)

Declared In
NSView.h

pageFooter

Returns a default footer string that includes the current page number and page count.

```
- (NSAttributedString *)pageFooter
```

Discussion
A printable view class can override this method to substitute its own content in place of the default value. You should not need to call this method directly. The printing system calls it once per page during printing.

Footers are generated only if the user defaults contain the key `NSPrintHeaderAndFooter` with the value `YES`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- pageHeader (page 3194)

Declared In
NSView.h

pageHeader

Returns a default header string that includes the print job title and date.

```
- (NSAttributedString *)pageHeader
```

Discussion
Typically, the print job title is the same as the window title. A printable view class can override this method to substitute its own content in place of the default value. You should not need to call this method directly. The printing system calls it once per page during printing.

Headers are generated only if the user defaults contain the key `NSPrintHeaderAndFooter` with the value `YES`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- pageFooter (page 3194)

Declared In
NSView.h

performKeyEquivalent:

Implemented by subclasses to respond to key equivalents (also known as keyboard shortcuts).

```
- (BOOL)performKeyEquivalent:(NSEvent *)theEvent
```

Parameters

theEvent

The key-down event object representing a key equivalent.

Return Value

YES if *theEvent* is a key equivalent that the receiver handled, NO if it is not a key equivalent that it should handle.

Discussion

If the receiver's key equivalent is the same as the characters of the key-down event *theEvent*, as returned by charactersIgnoringModifiers (page 1074), the receiver should take the appropriate action and return YES. Otherwise, it should return the result of invoking super's implementation. The default implementation of this method simply passes the message down the view hierarchy (from superviews to subviews) and returns NO if none of the receiver's subviews responds YES.

Availability

Available in Mac OS X v10.0 and later.

See Also

- performMnemonic: (page 3195)

keyDown: (page 3343) (NSWindow)

Declared In

NSView.h

performMnemonic:

Implemented by subclasses to respond to mnemonics.

```
- (BOOL)performMnemonic:(NSString *)aString
```

Parameters

aString

A string representing the mnemonic to handle.

Discussion

If the receiver's mnemonic is the same as the characters of the string *aString*, the receiver should take the appropriate action and return YES. Otherwise, it should return the result of invoking super's implementation. The default implementation of this method simply passes the message down the view hierarchy (from superviews to subviews) and returns NO if none of the receiver's subviews responds YES. Mnemonics are not supported in Mac OS X.

Availability

Available in Mac OS X v10.0 and later.

See Also

- performKeyEquivalent: (page 3195)

keyDown: (page 3343) (NSWindow)

Declared In
NSView.h

postsBoundsChangedNotifications

Returns YES if the receiver posts notifications to the default notification center whenever its bounds rectangle changes; returns NO otherwise.

```
- (BOOL)postsBoundsChangedNotifications
```

Discussion
See setPostsBoundsChangedNotifications: (page 3226) for a list of methods that result in notifications.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSView.h

postsFrameChangedNotifications

Returns YES if the receiver posts notifications to the default notification center whenever its frame rectangle changes; returns NO otherwise.

```
- (BOOL)postsFrameChangedNotifications
```

Discussion
See setPostsBoundsChangedNotifications: (page 3226) for a list of methods that result in notifications.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSView.h

preservesContentDuringLiveResize

Returns YES if the view supports the optimization of live-resize operations by preserving content that has not moved; otherwise, returns NO.

```
- (BOOL)preservesContentDuringLiveResize
```

Discussion
The default is NO. If your view supports the content preservation feature, you should override this method and have your implementation return YES.

Content preservation lets your view decide what to redraw during a live resize operation. If your view supports this feature, you should also provide a custom implementation of setFrameSize: (page 3221) that invalidates the portions of your view that actually need to be redrawn.

For information on how to implement this feature in your views, see *Cocoa Performance Guidelines*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setFrameSize:` (page 3221)

Declared In
`NSView.h`

previousKeyView

Returns the view object preceding the receiver in the key view loop.

```
- (NSView *)previousKeyView
```

Return Value
The view object preceding the receiver in the key view loop, or `nil` if there is none.

Discussion
This view should, if possible, be made first responder when the user navigates backward from the receiver using keyboard interface control.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `previousValidKeyView` (page 3197)
- `nextKeyView` (page 3192)
- `nextValidKeyView` (page 3193)
- `setNextKeyView:` (page 3226)

`selectNextKeyView:` (page 3364) (`NSWindow`)
`selectKeyViewFollowingView:` (page 3364) (`NSWindow`)
`selectPreviousKeyView:` (page 3365) (`NSWindow`)
`selectKeyViewPrecedingView:` (page 3364) (`NSWindow`)

Declared In
`NSView.h`

previousValidKeyView

Returns the closest view object in the key view loop that precedes the receiver and accepts first responder status.

```
- (NSView *)previousValidKeyView
```

Return Value
The closest view object in the key view loop that precedes the receiver and accepts first responder status, or `nil` if there is none.

Discussion
This method ignores hidden views when it determines the previous valid key view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `previousKeyView` (page 3197)
- `nextValidKeyView` (page 3193)
- `nextKeyView` (page 3192)
- `setNextKeyView:` (page 3226)

`selectNextKeyView:` (page 3364) (`NSWindow`)
`selectKeyViewFollowingView:` (page 3364) (`NSWindow`)
`selectPreviousKeyView:` (page 3365) (`NSWindow`)
`selectKeyViewPrecedingView:` (page 3364) (`NSWindow`)
- `setHidden:` (page 3222)

Declared In
`NSView.h`

print:

This action method opens the Print panel, and if the user chooses an option other than canceling, prints the receiver and all its subviews to the device specified in the Print panel.

```
- (void)print:(id)sender
```

Parameters
sender
The object that sent the message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `dataWithEPSInsideRect:` (page 3162)
- `writeEPSInsideRect:toPasteboard:` (page 3248)

Declared In
`NSView.h`

printJobTitle

Returns the receiver's print job title.

```
- (NSString *)printJobTitle
```

Discussion
The default implementation first tries the window's `NSDocument` display name (`displayName` (page 945)), then the window's title.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ImageApp

Declared In
NSView.h

rectForPage:

Implemented by subclasses to determine the portion of the receiver to be printed for the page number page.

```
- (NSRect)rectForPage:(NSInteger)pageNumber
```

Parameters

pageNumber

An integer indicating a page number. Page numbers are one-based—that is pages run from one to *N*.

Return Value
A rectangle defining the region of the receiver to be printed for *pageNumber*. This method returns NSZeroRect if *pageNumber* is outside the receiver's bounds.

Discussion
If the receiver responded YES to an earlier knowsPageRange: (page 3185) message, this method is invoked for each page it specified in the out parameters of that message. The receiver is later made to display this rectangle in order to generate the image for this page.

If an NSView object responds NO to knowsPageRange: (page 3185), this method isn't invoked by the printing mechanism.

Availability
Available in Mac OS X v10.0 and later.

See Also
- adjustPageHeightNew:top:bottom:limit: (page 3143)
- adjustPageWidthNew:left:right:limit: (page 3144)

Declared In
NSView.h

rectPreservedDuringLiveResize

Returns the rectangle identifying the portion of your view that did not change during a live resize operation.

```
- (NSRect)rectPreservedDuringLiveResize
```

Discussion
The returned rectangle is in the coordinate system of your view and reflects the space your view previously occupied. This rectangle may be smaller or the same size as your view's current bounds, depending on whether the view grew or shrunk.

If your view does not support content preservation during live resizing, the returned rectangle will be empty. To support content preservation, override preservesContentDuringLiveResize (page 3196) in your view and have your implementation return YES.

Note: The window containing your view must also support content preservation. To enable support for this feature in your window, use the `setPreservesContentDuringLiveResize:` (page 3393) method of `NSWindow`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `getRectsExposedDuringLiveResize:count:` (page 3177)
- `preservesContentDuringLiveResize` (page 3196)

Declared In
`NSView.h`

reflectScrolledClipView:

Notifies a clip view's superview that either the clip view's bounds rectangle or the document view's frame rectangle has changed, and that any indicators of the scroll position need to be adjusted.

```
- (void)reflectScrolledClipView:(NSClipView *)aClipView
```

Parameters
aClipView
The `NSClipView` object whose superview is to be notified.

Discussion
`NSScrollView` implements this method to update its `NSScroller` objects.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSClipView.h`

registeredDraggedTypes

Returns the array of pasteboard drag types that the view can accept.

```
- (NSArray *)registeredDraggedTypes
```

Discussion
This method returns the types registered by calling `registerForDraggedTypes:` (page 3201). Each element of the array is a uniform type identifier. The returned elements are in no particular order, but the array is guaranteed not to contain duplicate entries.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSView.h`

registerForDraggedTypes:

Registers the pasteboard types that the receiver will accept as the destination of an image-dragging session.

```
- (void)registerForDraggedTypes:(NSArray *)newTypes
```

Parameters

newTypes

An array of uniform type identifiers. See `Types for Standard Data (Mac OS X 10.6 and later)` (page 1904) for descriptions of the pasteboard type identifiers.

Discussion

Registering an `NSView` object for dragged types automatically makes it a candidate destination object for a dragging session. As such, it must properly implement some or all of the `NSDraggingDestination` protocol methods. As a convenience, `NSView` provides default implementations of these methods. See the `NSDraggingDestination` protocol specification for details.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `registeredDraggedTypes` (page 3200)
- `unregisterDraggedTypes` (page 3239)

Related Sample Code

CocoaDragAndDrop

CompositeLab

QTKitMovieShuffler

Declared In

`NSView.h`

releaseGState

Frees the receiver's graphics state object, if it has one.

```
- (void)releaseGState
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- `allocateGState` (page 3146)

Declared In

`NSView.h`

removeAllToolTips

Removes all tool tips assigned to the receiver.

```
- (void)removeAllToolTips
```

Discussion
This method operates on tool tips created using either `addToolTipRect:owner:userData:` (page 3141) or `setToolTip:` (page 3229).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

removeCursorRect:cursor:

Completely removes a cursor rectangle from the receiver.

```
- (void)removeCursorRect:(NSRect)aRect
    cursor:(NSCursor *)aCursor
```

Parameters

aRect

A rectangle defining a region of the receiver. Must match a value previously specified using `addCursorRect:cursor:` (page 3139).

aCursor

An object representing a cursor. Must match a value previously specified using `addCursorRect:cursor:` (page 3139).

Discussion
You should rarely need to use this method. `resetCursorRects` (page 3206), which is invoked any time cursor rectangles need to be rebuilt, should establish only the cursor rectangles needed. If you implement `resetCursorRects` (page 3206) in this way, you can then simply modify the state that `resetCursorRects` (page 3206) uses to build its cursor rectangles and then invoke the `NSWindow` method `invalidateCursorRectsForView:` (page 3335).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- discardCursorRects` (page 3163)

Declared In
`NSView.h`

removeFromSuperview

Unlinks the receiver from its superview and its window, removes it from the responder chain, and invalidates its cursor rectangles.

```
- (void)removeFromSuperview
```

Discussion
The receiver is also released; if you plan to reuse it, be sure to retain it before sending this message and to release it as appropriate when adding it as a subview of another `NSView`.

Never invoke this method during display.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addSubview:` (page 3139)
- `addSubview:positioned:relativeTo:` (page 3140)
- `removeFromSuperviewWithoutNeedingDisplay` (page 3203)

Related Sample Code
CocoaSlides
GLUT
Quartz Composer WWDC 2005 TextEdit
Sketch-112
SourceView

Declared In
`NSView.h`

removeFromSuperviewWithoutNeedingDisplay

Unlinks the receiver from its superview and its window and removes it from the responder chain, but does not invalidate its cursor rectangles to cause redrawing.

```
- (void)removeFromSuperviewWithoutNeedingDisplay
```

Discussion
The receiver is also released; if you plan to reuse it, be sure to retain it before sending this message and to release it as appropriate when adding it as a subview of another view.

Unlike its counterpart, `removeFromSuperview` (page 3202), this method can be safely invoked during display.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addSubview:` (page 3139)
- `addSubview:positioned:relativeTo:` (page 3140)

Related Sample Code
CoreRecipes

Declared In
`NSView.h`

removeToolTip:

Removes the tool tip identified by specified tag.

```
- (void)removeToolTip:(NSToolTipTag)tag
```

Parameters

tag

An integer tag that is the value returned by a previous `addToolTipRect:owner:userData:` (page 3141) message.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

removeTrackingArea:

Removes a given tracking area from the receiver.

```
- (void)removeTrackingArea:(NSTrackingArea *)trackingArea
```

Parameters

trackingArea

The tracking area to remove from the receiver.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
TrackIt

Declared In
`NSView.h`

removeTrackingRect:

Removes the tracking rectangle identified by a tag.

```
- (void)removeTrackingRect:(NSTrackingRectTag)aTag
```

Parameters

aTag

An integer value identifying a tracking rectangle. It was returned by a previously sent `addTrackingRect:owner:userData:assumeInside:` (page 3142) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addTrackingRect:owner:userData:assumeInside:` (page 3142)
- `removeTrackingArea:` (page 3204)

Related Sample Code
GLUT
ImageMap
ImageMapExample

Declared In
NSView.h

renewGState

Invalidates the receiver's graphics state object, if it has one.

```
- (void)renewGState
```

Discussion
The receiver's graphics state object will be regenerated using `setUpGState` (page 3229) the next time the receiver is focused for drawing

Availability
Available in Mac OS X v10.0 and later.

See Also
- `lockFocus` (page 3187)

Related Sample Code
GLSLShowpiece
VideoViewer

Declared In
NSView.h

replaceSubview:with:

Replaces one of the receiver's subviews with another view.

```
- (void)replaceSubview:(NSView *)oldView
    with:(NSView *)newView
```

Parameters
oldView
The view to be replaced by `newView`. May not be `nil`.

newView
The view to replace `oldView`. May not be `nil`.

Discussion
This method does nothing if *oldView* is not a subview of the receiver.

Neither *oldView* nor *newView* may be `nil`, and the behavior is undefined if either of these parameters is `nil`.

This method causes *oldView* to be released; if you plan to reuse it, be sure to retain it before sending this message and to release it as appropriate when adding it as a subview of another NSView.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `addSubview:` (page 3139)

`- addSubview:positioned:relativeTo:` (page 3140)

Related Sample Code
BasicCocoaAnimations
CocoaSlides
MatrixMixerTest

Declared In
`NSView.h`

resetCursorRects

Overridden by subclasses to define their default cursor rectangles.

```
- (void)resetCursorRects
```

Discussion
A subclass's implementation must invoke `addCursorRect:cursor:` (page 3139) for each cursor rectangle it wants to establish. The default implementation does nothing.

Application code should never invoke this method directly; it's invoked automatically as described in ""Responding to User Events and Actions" in *View Programming Guide*." Use the `invalidateCursorRectsForView:` (page 3335) method instead to explicitly rebuild cursor rectangles.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- visibleRect` (page 3245)

Related Sample Code
GeekGameBoard
GLUT
TextLinks

Declared In
`NSView.h`

resizeSubviewsWithOldSize:

Informs the receiver's subviews that the receiver's bounds rectangle size has changed.

```
- (void)resizeSubviewsWithOldSize:(NSSize)oldBoundsSize
```

Parameters
oldBoundsSize
The previous size of the receiver's bounds rectangle.

Discussion
If the receiver is configured to autoresize its subviews, this method is automatically invoked by any method that changes the receiver's frame size.

The default implementation sends `resizeWithOldSuperviewSize:` (page 3207) to the receiver's subviews with *oldBoundsSize* as the argument. You shouldn't invoke this method directly, but you can override it to define a specific retiling behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAutoresizesSubviews:` (page 3212)

Declared In
`NSView.h`

resizeWithOldSuperviewSize:

Informs the receiver that the bounds size of its superview has changed.

```
- (void)resizeWithOldSuperviewSize:(NSSize)oldBoundsSize
```

Parameters

oldBoundsSize
The previous size of the superview's bounds rectangle.

Discussion
This method is normally invoked automatically from `resizeSubviewsWithOldSize:` (page 3206).

The default implementation resizes the receiver according to the autoresizing options listed under the `setAutoresizingMask:` (page 3212) method description. You shouldn't invoke this method directly, but you can override it to define a specific resizing behavior.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

rightMouseDown:

Informs the receiver that the user has pressed the right mouse button.

```
- (void)rightMouseDown:(NSEvent *)theEvent
```

Parameters

theEvent
An object encapsulating information about the mouse-down event.

Discussion
The default implementation calls `menuForEvent:` (page 3189) and, if non `nil`, presents the contextual menu.

This behavior differs from other mouse events as the event is not passed up the responder chain.

See Also
- `rightMouseDown:` (page 2189) (NSResponder)

rotateByAngle:

Rotates the receiver's bounds rectangle by a specified degree value around the origin of the coordinate system, (0.0, 0.0).

```
- (void)rotateByAngle:(CGFloat)angle
```

Parameters

angle

A `float` value specifying the angle of rotation, in degrees.

Discussion

See the `setBoundsRotation:` (page 3215) method description for more information. This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 3163) or `setNeedsDisplay:` (page 3225).

This method posts an `NSViewBoundsDidChangeNotification` (page 3256) to the default notification center if the receiver is configured to do so.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setFrameRotation:` (page 3221)
- `setPostsBoundsChangedNotifications:` (page 3226)

Declared In

`NSView.h`

scaleUnitSquareToSize:

Scales the receiver's coordinate system so that the unit square scales to the specified dimensions.

```
- (void)scaleUnitSquareToSize:(NSSize)newUnitSize
```

Parameters

newUnitSize

An `NSSize` structure specifying the new unit size.

Discussion

For example, a *newUnitSize* of (0.5, 1.0) causes the receiver's horizontal coordinates to be halved, in turn doubling the width of its bounds rectangle. Note that scaling is performed from the origin of the coordinate system, (0.0, 0.0), not the origin of the bounds rectangle; as a result, both the origin and size of the bounds rectangle are changed. The frame rectangle remains unchanged.

This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 3163) or `setNeedsDisplay:` (page 3225).

This method posts an `NSViewBoundsDidChangeNotification` (page 3256) to the default notification center if the receiver is configured to do so.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `setBoundsSize:` (page 3216)
- `setPostsBoundsChangedNotifications:` (page 3226)

Declared In
`NSView.h`

scrollClipView:toPoint:

Notifies the superview of a clip view that the clip view needs to reset the origin of its bounds rectangle.

```
- (void)scrollClipView:(NSClipView *)aClipView
    toPoint:(NSPoint)newOrigin
```

Parameters

aClipView
The `NSClipView` object whose superview is to be notified.

newOrigin
A point that specifies the new origin of the clip view's bounds rectangle.

Discussion
The superview of *aClipView* should then send a `scrollToPoint:` (page 635) message to *aClipView* with *newOrigin* as the argument. This mechanism is provided so the `NSClipView` object's superview can coordinate scrolling of multiple tiled clip views.

Availability
Available in Mac OS X v10.0 and later.

See Also
`scrollToPoint:` (page 635) (NSClipView)

Declared In
`NSClipView.h`

scrollPoint:

Scrolls the receiver's closest ancestor `NSClipView` object so a point in the receiver lies at the origin of the clip view's bounds rectangle.

```
- (void)scrollPoint:(NSPoint)aPoint
```

Parameters

aPoint
The point in the receiver to scroll to.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `autoscroll:` (page 3148)

`scrollToPoint:` (page 635) (`NSClipView`)
- `isDescendantOf:` (page 3181)

Related Sample Code
GLUT

Declared In
`NSView.h`

scrollRect:by:

Copies the visible portion of the receiver's rendered image within a region and lays that portion down again at a specified offset .

```
- (void)scrollRect:(NSRect)aRect
    by:(NSSize)offset
```

Parameters

aRect

A rectangle defining a region of the receiver.

offset

A `NSSize` structure that specifies an offset from from *aRect*'s origin.

Discussion
This method is useful during scrolling or translation of the coordinate system to efficiently move as much of the receiver's rendered image as possible without requiring it to be redrawn, following these steps:

1. Invoke `scrollRect:by:` (page 3210) to copy the rendered image.
2. Move the view object's origin or scroll it within its superview.
3. Calculate the newly exposed rectangles and invoke either `setNeedsDisplay:` (page 3225) or `setNeedsDisplayInRect:` (page 3225) to draw them.

You should rarely need to use this method, however. The `scrollPoint:` (page 3209), `scrollRectToVisible:` (page 3210), and `autoscroll:` (page 3148) methods automatically perform optimized scrolling.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBoundsOrigin:` (page 3214)
- `translateOriginToPoint:` (page 3237)

Declared In
`NSView.h`

scrollRectToVisible:

Scrolls the receiver's closest ancestor `NSClipView` object the minimum distance needed so a specified region of the receiver becomes visible in the clip view.

```
- (BOOL)scrollRectToVisible:(NSRect)aRect
```

Parameters

aRect

The rectangle to be made visible in the clip view.

Discussion

YES if any scrolling is performed; otherwise returns NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- autoscroll: (page 3148)
scrollToPoint: (page 635) (NSClipView)
- isDescendantOf: (page 3181)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

NSView.h

setAcceptsTouchEvents:

Sets whether the view should accept touch events.

```
- (void)setAcceptsTouchEvents:(BOOL)flag
```

Parameters

flag

YES if the view should accept touch events, otherwise NO.

Discussion

By default views do not accept touch events.

Availability

Available in Mac OS X v10.6 and later.

See Also

- acceptsTouchEvents (page 3138)

Declared In

NSView.h

setAlphaValue:

Sets the opacity of the receiver.

```
- (void)setAlphaValue:(CGFloat)viewAlpha
```

Parameters

viewAlpha

The desired opacity of the receiver.

Discussion
This method sets the value of the `opacity` property of the receiver's layer. Possible values are between 0.0 (transparent) and 1.0 (opaque). The default is 1.0.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
LayerBackedOpenGLView

Declared In
`NSView.h`

setAutoresizesSubviews:

Determines whether the receiver automatically resizes its subviews when its frame size changes.

```
- (void)setAutoresizesSubviews:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver invokes `resizeSubviewsWithOldSize:` (page 3206) whenever its frame size changes; if `NO`, it doesn't.

Discussion
View objects do autoresize their subviews by default.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- autoresizesSubviews` (page 3147)

Related Sample Code
FunHouse
GLUT
Sproing
TextSizingExample

Declared In
`NSView.h`

setAutoresizingMask:

Determines how the receiver's `resizeWithOldSuperviewSize:` (page 3207) method changes its frame rectangle.

```
- (void)setAutoresizingMask:(NSUInteger)mask
```

Parameters

mask

An integer bit mask. *mask* can be specified by combining using the C bitwise OR operator any of the options described in "Resizing masks" (page 3250).

Discussion

Where more than one option along an axis is set, `resizeWithOldSuperviewSize:` (page 3207) by default distributes the size difference as evenly as possible among the flexible portions. For example, if `NSViewWidthSizable` and `NSViewMaxXMargin` are set and the superview's width has increased by 10.0 units, the receiver's frame and right margin are each widened by 5.0 units.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `autoresizingMask` (page 3147)
- `resizeSubviewsWithOldSize:` (page 3206)
- `setAutoresizesSubviews:` (page 3212)

Related Sample Code

FunHouse
PhotoSearch
PredicateEditorSample
Quartz Composer QCTV
TextSizingExample

Declared In

`NSView.h`

setBackgroundFilters:

An array of CoreImage filters that are applied to the receiver's background.

```
- (void)setBackgroundFilters:(NSArray *)filters
```

Parameters

filters

An array of CoreImage filters.

Discussion

This method sets the value of the `backgroundFilters` property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

LayerBackedOpenGLView

Declared In

`NSView.h`

setBounds:

Sets the receiver's bounds rectangle.

```
- (void)setBounds:(NSRect)boundsRect
```

Parameters

boundsRect

A rectangle defining the new bounds of the receiver.

Discussion

The bounds rectangle determines the origin and scale of the receiver's coordinate system within its frame rectangle. This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 3163) or `setNeedsDisplay:` (page 3225).

This method posts an `NSViewBoundsDidChangeNotification` (page 3256) to the default notification center if the receiver is configured to do so.

After calling this method, `NSView` creates an internal transform (or appends these changes to an existing internal transform) to convert from frame coordinates to bounds coordinates in your view. As long as the width-to-height ratio of the two coordinate systems remains the same, your content appears normal. If the ratios differ, your content may appear skewed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `bounds` (page 3150)
- `setBoundsRotation:` (page 3215)
- `setBoundsOrigin:` (page 3214)
- `setBoundsSize:` (page 3216)
- `setFrame:` (page 3218)
- `setPostsBoundsChangedNotifications:` (page 3226)

Related Sample Code

CIAnnotation

Declared In

`NSView.h`

setBoundsOrigin:

Sets the origin of the receiver's bounds rectangle to a specified point,

```
- (void)setBoundsOrigin:(NSPoint)newOrigin
```

Parameters

newOrigin

A point specifying the new bounds origin of the receiver.

Discussion

In setting the new bounds origin, this method effectively shifts the receiver's coordinate system so *newOrigin* lies at the origin of the receiver's frame rectangle. It neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` or `setNeedsDisplay:` (page 3225).

This method posts an `NSViewBoundsDidChangeNotification` (page 3256) to the default notification center if the receiver is configured to do so.

After calling this method, `NSView` creates an internal transform (or appends these changes to an existing internal transform) to convert from frame coordinates to bounds coordinates in your view. As long as the width-to-height ratio of the two coordinate systems remains the same, your content appears normal. If the ratios differ, your content may appear skewed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `translateOriginToPoint:` (page 3237)
- `bounds` (page 3150)
- `setBoundsRotation:` (page 3215)
- `setBounds:` (page 3214)
- `setBoundsSize:` (page 3216)
- `setPostsBoundsChangedNotifications:` (page 3226)

Related Sample Code
Polygons
Rulers

Declared In
`NSView.h`

setBoundsRotation:

Sets the rotation of the receiver's bounds rectangle to a specific degree value.

```
- (void)setBoundsRotation:(CGFloat)angle
```

Parameters

angle
A `float` value specifying the angle of rotation, in degrees.

Discussion
Positive values indicate counterclockwise rotation, negative clockwise. Rotation is performed around the coordinate system origin, (0.0, 0.0), which need not coincide with that of the frame rectangle or the bounds rectangle. This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 3163) or `setNeedsDisplay:` (page 3225).

This method posts an `NSViewBoundsDidChangeNotification` (page 3256) to the default notification center if the receiver is configured to do so.

Bounds rotation affects the orientation of the drawing within the view object's frame rectangle, but not the orientation of the frame rectangle itself. Also, for a rotated bounds rectangle to enclose all the visible areas of its view object—that is, to guarantee coverage over the frame rectangle—it must also contain some areas that aren't visible. This can cause unnecessary drawing to be requested, which may affect performance. It may be better in many cases to rotate the coordinate system in the `drawRect:` (page 3170) method rather than use this method.

After calling this method, NSView creates an internal transform (or appends these changes to an existing internal transform) to convert from frame coordinates to bounds coordinates in your view. As long as the width-to-height ratio of the two coordinate systems remains the same, your content appears normal. If the ratios differ, your content may appear skewed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- rotateByAngle: (page 3208)
- boundsRotation (page 3151)
- setFrameRotation: (page 3221)
- setPostsBoundsChangedNotifications: (page 3226)

Related Sample Code
TrackIt

Declared In
NSView.h

setBoundsSize:

Sets the size of the receiver's bounds rectangle to specified dimensions, inversely scaling its coordinate system relative to its frame rectangle.

```
- (void)setBoundsSize:(NSSize)newSize
```

Parameters

newSize
An NSSize structure specifying the new width and height of the receiver's bounds rectangle.

Discussion
For example, a view object with a frame size of (100.0, 100.0) and a bounds size of (200.0, 100.0) draws half as wide along the x axis. This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with display (page 3163) or setNeedsDisplay: (page 3225).

This method posts an NSViewBoundsDidChangeNotification (page 3256) to the default notification center if the receiver is configured to do so.

After calling this method, NSView creates an internal transform (or appends these changes to an existing internal transform) to convert from frame coordinates to bounds coordinates in your view. As long as the width-to-height ratio of the two coordinate systems remains the same, your content appears normal. If the ratios differ, your content may appear skewed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- bounds (page 3150)
- setBoundsRotation: (page 3215)
- setBounds: (page 3214)
- setBoundsOrigin: (page 3214)
- setPostsBoundsChangedNotifications: (page 3226)

Related Sample Code
ObjectPath
Polygons
Quartz Composer WWDC 2005 TextEdit
Rulers
Sketch+Accessibility

Declared In
`NSView.h`

setCanDrawConcurrently:

Sets whether the view's `drawRect:` method can be invoked on a background thread.

```
- (void)setCanDrawConcurrently:(BOOL)flag
```

Parameters

flag

`YES` if `drawRect:` (page 3170) can be invoked from a background thread, otherwise `NO`. The default is `NO` for most types of views..

Discussion
The view's window must also have its `allowsConcurrentViewDrawing` (page 3298) property set to `YES` (the default) for threading of view drawing to actually take place.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setCanDrawConcurrently:` (page 3217)
`allowsConcurrentViewDrawing` (page 3298) (NSWindow)

Declared In
`NSView.h`

setCompositingFilter:

Sets a CoreImage filter that is used to composite the receiver's contents with the background.

```
- (void)setCompositingFilter:(CIFilter *)filter
```

Parameters

filter

A CoreImage filter.

Discussion
This method sets the value of the `compositingFilter` (page 3154) property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSView.h

setContentFilters:

Sets the array of CoreImage filters that are applied to the contents of the receiver and its sublayers.

```
- (void)setContentFilters:(NSArray *)filters
```

Parameters

filters

An array of CoreImage filters.

Discussion
This method sets the value of the `filters` property of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSView.h

setFocusRingType:

Sets the type of focus ring to be drawn around the receiver.

```
- (void)setFocusRingType:(NSFocusRingType)focusRingType
```

Parameters

focusRingType

An `enum` constant identifying a type of focus ring. Possible values are listed in `NSFocusRingType` (page 4009). You can specify `NSFocusRingTypeNone` to indicate you do not want your view to have a focus ring.

Discussion
This method only sets the desired focus ring type and does not cause the view to draw the actual focus ring. You are responsible for drawing the focus ring in your view's drawRect: (page 3170) method whenever your view is made the first responder.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `focusRingType` (page 3175)

Declared In
NSView.h

setFrame:

Sets the receiver's frame rectangle to the specified rectangle.

```
- (void)setFrame:(NSRect)frameRect
```

Parameters

frameRect

The new frame rectangle for the view.

Discussion

This method, in setting the frame rectangle, repositions and resizes the receiver within the coordinate system of its superview. It neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 3163) or `setNeedsDisplay:` (page 3225).

This method posts an `NSViewFrameDidChangeNotification` (page 3257) to the default notification center if the receiver is configured to do so.

If your view does not use a custom bounds rectangle, this method also sets your view bounds to match the size of the new frame. You specify a custom bounds rectangle by calling `setBounds:` (page 3214), `setBoundsOrigin:` (page 3214), `setBoundsRotation:` (page 3215), or `setBoundsSize:` (page 3216)explicitly. Once set, NSView creates an internal transform to convert from frame coordinates to bounds coordinates. As long as the width-to-height ratio of the two coordinate systems remains the same, your content appears normal. If the ratios differ, your content may appear skewed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `frame` (page 3175)
- `setFrameRotation:` (page 3221)
- `setFrameOrigin:` (page 3220)
- `setFrameSize:` (page 3221)
- `setPostsFrameChangedNotifications:` (page 3227)

Related Sample Code

FunHouse
Quartz Composer WWDC 2005 TextEdit
Rulers
Sproing
TipWrapper

Declared In

`NSView.h`

setFrameCenterRotation:

Rotates the frame of the receiver about the layer's position.

```
- (void)setFrameCenterRotation:(CGFloat)angle
```

Parameters

angle

The angle to rotate the frame around the center of the receiver.

Discussion
If the application has altered the layer's `anchorPoint` property, the behavior is undefined. Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

setFrameOrigin:

Sets the origin of the receiver's frame rectangle to the specified point, effectively repositioning it within its superview.

```
- (void)setFrameOrigin:(NSPoint)newOrigin
```

Parameters

newOrigin
The point that is the new origin of the receiver's frame.

Discussion
This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 3163) or `setNeedsDisplay:` (page 3225).

This method posts an `NSViewFrameDidChangeNotification` (page 3257) to the default notification center if the receiver is configured to do so.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `frame` (page 3175)
- `setFrameSize:` (page 3221)
- `setFrame:` (page 3218)
- `setFrameRotation:` (page 3221)
- `setPostsFrameChangedNotifications:` (page 3227)

Related Sample Code
CocoaCreateMovie
CoreRecipes
Reducer
TextSizingExample
WhackedTV

Declared In
`NSView.h`

setFrameRotation:

Sets the rotation of the receiver's frame rectangle to a specified degree value, rotating it within its superview without affecting its coordinate system.

```
- (void)setFrameRotation:(CGFloat)angle
```

Parameters

angle

A `float` value indicating the degree of rotation.

Discussion

Positive values indicate counterclockwise rotation, negative clockwise. Rotation is performed around the origin of the frame rectangle.

This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 3163) or `setNeedsDisplay:` (page 3225).

This method posts an `NSViewFrameDidChangeNotification` (page 3257) to the default notification center if the receiver is configured to do so.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `frameRotation` (page 3176)
- `setBoundsRotation:` (page 3215)

Declared In

`NSView.h`

setFrameSize:

Sets the size of the receiver's frame rectangle to the specified dimensions, resizing it within its superview without affecting its coordinate system.

```
- (void)setFrameSize:(NSSize)newSize
```

Parameters

newSize

An `NSSize` structure specifying the new height and width of the frame rectangle.

Discussion

This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 3163) or `setNeedsDisplay:` (page 3225).

This method posts an `NSViewFrameDidChangeNotification` (page 3257) to the default notification center if the receiver is configured to do so.

In Mac OS X version 10.4 and later, you can override this method to support content preservation during live resizing. In your overridden implementation, include some conditional code to be executed only during a live resize operation. Your code must invalidate any portions of your view that need to be redrawn.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `frame` (page 3175)
- `setFrameOrigin:` (page 3220)
- `setFrame:` (page 3218)
- `setFrameRotation:` (page 3221)
- `setPostsFrameChangedNotifications:` (page 3227)

Related Sample Code
CocoaCreateMovie
GLUT
MyPhoto
Quartz Composer WWDC 2005 TextEdit
WhackedTV

Declared In
`NSView.h`

setHidden:

Sets whether the view is hidden.

```
- (void)setHidden:(BOOL)flag
```

Parameters
flag
`YES` if the receiver is to be hidden, `NO` otherwise.

Discussion
A hidden view disappears from its window and does not receive input events. It remains in its superview's list of subviews, however, and participates in autoresizing as usual. The Application Kit also disables any cursor rectangle, tool-tip rectangle, or tracking rectangle associated with a hidden view. Hiding a view with subviews has the effect of hiding those subviews and any view descendants they might have. This effect is implicit and does not alter the hidden state of the receiver's descendants as reported by `isHidden` (page 3182).

Hiding the view that is the window's current first responder causes the view's next valid key view (`nextValidKeyView` (page 3193)) to become the new first responder. A hidden view remains in the `nextKeyView` (page 3192) chain of views it was previously part of, but is ignored during keyboard navigation.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `isHidden` (page 3182)
- `isHiddenOrHasHiddenAncestor` (page 3182)

Related Sample Code
DatePicker
FinalCutPro_AppleEvents
IdentitySample
ImageClient
Quartz 2D Transformer

Declared In
NSView.h

setKeyboardFocusRingNeedsDisplayInRect:

Invalidates the area around the focus ring.

```
- (void)setKeyboardFocusRingNeedsDisplayInRect:(NSRect)rect
```

Parameters

rect

The rectangle of the control or cell defining the area around the focus ring. *rect* will be expanded to include the focus ring for invalidation.

Availability
Available in Mac OS X v10.1 and later.

Declared In
NSView.h

setLayer:

Sets the Core Animation layer that the receiver uses for layer-backing to the specified layer.

```
- (void)setLayer:(CALayer *)newLayer
```

Parameters

newLayer

A Core Animation layer to use as the receiver's backing store.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
CocoaSlides
CoreAnimationKioskStyleMenu

Declared In
NSView.h

setLayerContentsPlacement:

Sets the view's layer contents placement policy.

```
- (void)setLayerContentsPlacement:(NSViewLayerContentsPlacement)newPlacement
```

Parameters

newPlacement

The placement policy. See "NSViewLayerContentsPlacement" (page 3253) for supported values.

Discussion
The content placement determines how the backing layer's existing cached content image will be mapped into the layer as the layer is resized. It is analogous to, and underpinned by, the CALayer class `contentsGravity` property. See `setLayerContentsRedrawPolicy:` (page 3224) for more information.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- layerContentsPlacement` (page 3186)

Declared In
`NSView.h`

setLayerContentsRedrawPolicy:

Sets the receiver layer contents redraw policy.

```
- (void)setLayerContentsRedrawPolicy:(NSViewLayerContentsRedrawPolicy)newPolicy
```

Parameters
newPolicy
The redraw policy. See "NSViewLayerContentsRedrawPolicy" (page 3252) for supported values.

Discussion
If you know that redrawing at each animation frame is not necessary to produce correctly rendered results for a particular view, or are willing to accept an approximation of the view's intermediate appearance during potentially brief animations in exchange for an animation performance and smoothness benefit, you can change the view's `layerContentsRedrawPolicy` (page 3186) to one of the modes that does not require constant redrawing.

When doing this, you must also specify the desired layer content placement for the view. The content placement determines how the backing layer's existing cached content image will be mapped into the layer as the layer is resized. It is analogous to, and underpinned by, the CALayer class `contentsGravity` property.

For a view that has no associated layer, or that has been assigned a developer-provided layer (a layer-hosting view) using the NSView `setLayer:` (page 3223) method, the default contents redraw policy is is `NSViewLayerContentsRedrawNever` (page 3253), with an accompanying `layerContentsPlacement` (page 3186) of `NSViewLayerContentsPlacementScaleAxesIndependently` (page 3254). This instructs AppKit that it is not allowed to replace the layer's content, and provides the same content placement as CALayer's default `contentsGravity` setting of `kCAGravityResize`. For a view that has acquired an AppKit-generated backing layer (a layer-backed view), AppKit sets the contents redraw policy to a default of `NSViewLayerContentsRedrawDuringViewResize` (page 3253), forcing the view's content to be continually redrawn into the view's backing layer during animated resizing of the view, which produces strictly correct but not optimally performance results.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSView.h`

setNeedsDisplay:

Controls whether the receiver's entire bounds is marked as needing display.

```
- (void)setNeedsDisplay:(BOOL)flag
```

Parameters

flag

If YES, marks the receiver's entire bounds as needing display; if NO, marks it as not needing display.

Discussion

Whenever the data or state used for drawing a view object changes, the view should be sent a setNeedsDisplay: message. NSView objects marked as needing display are automatically redisplayed on each pass through the application's event loop. (View objects that need to redisplay before the event loop comes around can of course immediately be sent the appropriate display... method.)

Availability

Available in Mac OS X v10.0 and later.

See Also

- setNeedsDisplayInRect: (page 3225)
- needsDisplay (page 3191)

Related Sample Code

FunHouse
OpenALExample
Quartz 2D Shadings
VertexPerformanceTest
WhackedTV

Declared In

NSView.h

setNeedsDisplayInRect:

Marks the region of the receiver within the specified rectangle as needing display, increasing the receiver's existing invalid region to include it.

```
- (void)setNeedsDisplayInRect:(NSRect)invalidRect
```

Parameters

invalidRect

The rectangular region of the receiver to mark as invalid; it should be specified in the coordinate system of the receiver.

Discussion

A later displayIfNeeded... method will then perform drawing only within the invalid region. View objects marked as needing display are automatically redisplayed on each pass through the application's event loop. (View objects that need to redisplay before the event loop comes around can of course immediately be sent the appropriate display... method.)

Availability

Available in Mac OS X v10.0 and later.

See Also
- `setNeedsDisplay:` (page 3225)
- `needsDisplay` (page 3191)

Related Sample Code
DragItemAround
JSPong
Rulers
Sketch-112
TextSizingExample

Declared In
`NSView.h`

setNextKeyView:

Inserts a specified view object after the receiver in the key view loop of the receiver's window.

```
- (void)setNextKeyView:(NSView *)aView
```

Parameters

aView

The `NSView` object to insert.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `nextKeyView` (page 3192)
- `nextValidKeyView` (page 3193)
- `previousKeyView` (page 3197)
- `previousValidKeyView` (page 3197)

Declared In
`NSView.h`

setPostsBoundsChangedNotifications:

Controls whether the receiver informs observers when its bounds rectangle changes.

```
- (void)setPostsBoundsChangedNotifications:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver will post notifications to the default notification center whenever its bounds rectangle changes; if *flag* is `NO` it won't.

Discussion
Note that if *flag* is `YES` and bounds notifications are suppressed, when the bounds change notification is reenabled the view will immediately post a single such notification if its bounds changed during this time. This will happen even if there has been no net change in the view's bounds.

The following methods can result in notification posting:

- `setBounds:` (page 3214)
- `setBoundsOrigin:` (page 3214)
- `setBoundsRotation:` (page 3215)
- `setBoundsSize:` (page 3216)
- `translateOriginToPoint:` (page 3237)
- `scaleUnitSquareToSize:` (page 3208)
- `rotateByAngle:` (page 3208)

Availability
Available in Mac OS X v10.0 and later.

See Also
- `postsBoundsChangedNotifications` (page 3196)

Related Sample Code
GLUT

Declared In
`NSView.h`

setPostsFrameChangedNotifications:

Controls whether the receiver informs observers when its frame rectangle changes.

```
- (void)setPostsFrameChangedNotifications:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver will post notifications to the default notification center whenever its frame rectangle changes; if *flag* is `NO` it won't.

Discussion
Note that if *flag* is `YES` and frame notifications are suppressed, when the frame change notification is reenabled the view will immediately post a single such notification if its frame changed during this time. This will happen even if there has been no net change in the view's frame.

The following methods can result in notification posting:

- `setFrame:` (page 3218)
- `setFrameOrigin:` (page 3220)
- `setFrameRotation:` (page 3221)
- `setFrameSize:` (page 3221)

Availability
Available in Mac OS X v10.0 and later.

See Also
- `postsFrameChangedNotifications` (page 3196)

Related Sample Code
GLUT

Declared In
`NSView.h`

setShadow:

Sets the shadow drawn by the receiver.

```
- (void)setShadow:(NSShadow *)shadow
```

Parameters

shadow
An instance of `NSShadow`.

Discussion
This method sets the `shadowColor`,`shadowOffset`, `shadowOpacity` and `shadowRadius` properties of the receiver's layer.

Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

setSubviews:

Sets the receiver's subviews to the specified subviews.

```
- (void)setSubviews:(NSArray *)newSubviews
```

Parameters

newSubviews
An array of subviews. The *newSubviews* array can consist of existing subviews of the receiver or other views that have `nil` as their superview. If *newSubviews* is `nil`, or contains duplicated views, or if any of its members have a superview other than `nil` or the receiver, an invalid argument exception is thrown.

Discussion
Using this method you can: reorder the receiver's existing subviews, add or remove subviews en masse, replace all of the receiver's subviews with a new set of subviews, or remove all the receiver's subviews.

Given a valid array of views in `newSubviews`, `setSubviews:` (page 3228) performs any required sorting of the subviews array, as well as sending any `addSubview:` (page 3139) and `removeFromSuperview` (page 3202) messages as necessary to leave the receiver with the requested new array of subviews. Any member of *newSubviews* that isn't already a subview of the receiver is added. Any member of the view's existing `subviews` array that isn't in *newSubviews* is removed. And any views that are in both `subviews` (page 3235) and *newSubviews* are moved in the subviews array as needed, without being removed and re-added.

This method marks the affected view and window areas as needing display.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
CocoaSlides

Declared In
NSView.h

setToolTip:

Sets the tool tip text for the view to *string*.

```
- (void)setToolTip:(NSString *)string
```

Parameters

string
A string that contains the text to use for the tool tip. If nil, it cancels tool tip display for the view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- toolTip (page 3236)

Related Sample Code
EnhancedAudioBurn

Declared In
NSView.h

setUpGState

Overridden by subclasses to (re)initialize the receiver's graphics state object.

```
- (void)setUpGState
```

Discussion
This method is automatically invoked when the graphics state object created using allocateGState (page 3146) needs to be initialized. The default implementation does nothing. Your subclass can override it to set the current font, line width, or any other graphics state parameter except coordinate transformations and the clipping path—these are established by the frame and bounds rectangles and by methods such as scaleUnitSquareToSize: (page 3208) and translateOriginToPoint: (page 3237). Note that drawRect: can further transform the coordinate system and clipping path for whatever temporary effects it needs.

Availability
Available in Mac OS X v10.0 and later.

See Also
- allocateGState (page 3146)
- renewGState (page 3205)

Declared In
NSView.h

setWantsLayer:

Specifies whether the receiver and its subviews use a Core Animation layer as a backing store.

```
- (void)setWantsLayer:(BOOL)flag
```

Parameters

flag

YES if the receiver and its subviews should use a Core Animation layer as its backing store, otherwise NO.

Discussion
The order that setWantsLayer: and setLayer: are called is important, it makes the distinction between a layer-backed view, and a layer-hosting view.

A layer-backed view is a view that is backed by a Core Animation layer. Any drawing done by the view is the cached in the backing layer. You configured a layer-backed view by simply invoking `setWantsLayer:` (page 3230) with a value of YES. The view class will automatically create the a backing layer for you, and you use the view class's drawing mechanisms. **When using layer-backed views you should never interact directly with the layer.**

A layer-hosting view is a view that contains a Core Animation layer that you intend to manipulate directly. You create a layer-hosting view by instantiating an instance of a Core Animation layer class and setting that layer using the view's `setLayer:` (page 3223) method. After doing so, you then invoke `setWantsLayer:` (page 3230) with a value of YES. **When using a layer-hosting view you should not rely on the view for drawing, nor should you add subviews to the layer-hosting view.**

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
AnimatedTableView
CocoaSlides
CoreAnimationKioskStyleMenu
LayerBackedOpenGLView

Declared In
NSView.h

setWantsRestingTouches:

Sets whether the view wants to receive resting touch events.

```
- (void)setWantsRestingTouches:(BOOL)flag
```

Parameters

flag

YES if the view wants resting touches, otherwise NO. The default is NO.

Discussion
A resting touch occurs when a user rests their thumb on a device (for example, the glass trackpad of a MacBook).

By default, these touches are not delivered and are not included in the event's set of touches. Touches may transition in and out of resting at any time. Unless the view wants restingTouches, began / ended events are simulated as touches transition from resting to active and vice versa.

In general resting touches should be ignored.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- wantsRestingTouches` (page 3247)

Declared In
`NSView.h`

shadow

Returns the shadow drawn by the receiver

```
- (NSShadow *)shadow
```

Return Value
An instance of `NSShadow` that is created using the `shadowColor`,`shadowOffset`, `shadowOpacity` and `shadowRadius` properties of the receiver's layer.

Discussion
Sending this message to a view that is not managing a Core Animation layer causes an exception.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
CocoaSlides

Declared In
`NSView.h`

shouldDelayWindowOrderingForEvent:

Overridden by subclasses to allow the user to drag images from the receiver without its window moving forward and possibly obscuring the destination and without activating the application.

```
- (BOOL)shouldDelayWindowOrderingForEvent:(NSEvent *)theEvent
```

Parameters
`theEvent`
An object representing an initial mouse-down event.

Return Value
If this method returns `YES`, the normal window-ordering and activation mechanism is delayed (not necessarily prevented) until the next mouse-up event. If it returns `NO`, then normal ordering and activation occur.

Discussion
Never invoke this method directly; it's invoked automatically for each mouse-down event directed at the NSView.

An `NSView` subclass that allows dragging should implement this method to return `YES` if *`theEvent`* is potentially the beginning of a dragging session or of some other context where window ordering isn't appropriate. This method is invoked before a `mouseDown:` (page 2164) message for *`theEvent`* is sent. The default implementation returns `NO`.

If, after delaying window ordering, the receiver actually initiates a dragging session or similar operation, it should also send a `preventWindowOrdering` (page 158) message to `NSApp`, which completely prevents the window from ordering forward and the activation from becoming active. `preventWindowOrdering` (page 158) is sent automatically by the `NSViewdragImage:...` and `dragFile:...` methods.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

shouldDrawColor

Returns `NO` if the receiver is being drawn in an `NSWindow` object (as opposed, for example, to being printed) and the window object can't store color; otherwise returns `YES`.

```
- (BOOL)shouldDrawColor
```

Discussion
A view object can base its drawing behavior on the return value of this method to improve its appearance in grayscale windows.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- drawRect:` (page 3170)
`canStoreColor` (page 3306) (NSWindow)

Declared In
`NSView.h`

showDefinitionForAttributedString:atPoint:

Shows a window displaying the definition of the of the attributed string at the specified point.

```
- (void)showDefinitionForAttributedString:(NSAttributedString *)attrString
    atPoint:(NSPoint)textBaselineOrigin
```

Parameters

attrString

The attributed string for which to show the definition. If the receiver is an instance of NSTextView, the `attrString` can be `nil`, in which case the text view will automatically supply values suitable for displaying definitions for the specified range within its text content.

textBaselineOrigin

Specifies the baseline origin of *attrString* in the receiver's view coordinate system.

Discussion

Shows a window that displays the definition (or other subject depending on available dictionaries) of the specified attributed string.

This method can be used for implementing the same functionality as the `NSTextView` "Look Up in Dictionary" contextual menu on a custom view.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- showDefinitionForAttributedString:range:options:baselineOriginProvider:` (page 3233)

Declared In

`NSView.h`

showDefinitionForAttributedString:range:options:baselineOriginProvider:

Shows a window displaying the definition of the specified range of the attributed string.

```
- (void)showDefinitionForAttributedString:(NSAttributedString *)attrString
    range:(NSRange)targetRange
    options:(NSDictionary *)options
    baselineOriginProvider:(NSPoint (^)(NSRange adjustedRange))originProvider
```

Parameters

attrString

The attributed string for which to show the definition. If the receiver is an instance of NSTextView, the `attrString` can be `nil`, in which case the text view will automatically supply values suitable for displaying definitions for the specified range within its text content.

range

The range of the attributed string to define. You can pass a zero-length range and the appropriate range will be auto-detected around the range's offset. That's the recommended approach when there is no selection.

options

An optional dictionary that specifies how the definition is displayed. See "NSDefinition Presentation Constants" (page 3255) for the key and it's possible values.

originProvider

The originProvider block object should return the baseline origin for the first character at the adjusted range.

If the receiver is an instance of NSTextView, the originProvider can be `NULL`, in which case the text view will automatically supply values suitable for displaying definitions for the specified range within its text content.

The block object takes a single argument:

adjustedRange

The adjusted range.

The block object returns an `NSPoint` to be used as the baseline origin of the first character, in the receiver view coordinate system.

Discussion
This method does not cause scrolling, so clients should perform any necessary scrolling before calling this method.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- showDefinitionForAttributedString:atPoint:` (page 3232)

Declared In
`NSView.h`

sortSubviewsUsingFunction:context:

Orders the receiver's immediate subviews using the specified comparator function.

```
- (void)sortSubviewsUsingFunction:(NSComparisonResult (*)(id, id, void *))compare
    context:(void *)context
```

Parameters

compare

A pointer to the comparator function. This function must take as arguments two subviews to be ordered and contextual data (supplied in *context* which may be arbitrary data used to help in the comparison. The comparator function should return `NSOrderedAscending` if the first subview should be ordered lower, `NSOrderedDescending` if the second subview should be ordered lower, and `NSOrderedSame` if their ordering isn't important.

context

Arbitrary data that might help the comparator function `compare` in its decisions.

Availability
Available in Mac OS X v10.0 and later.

See Also
`sortedArrayUsingFunction:context:` (`NSArray`)

Declared In
`NSView.h`

subviews

Return the receiver's immediate subviews.

```
- (NSArray *)subviews
```

Return Value
Returns an array containing the receiver's subviews.

Discussion
The order of the subviews may be considered as being back-to-front, but this does not imply invalidation and drawing behavior. The order is based on the order of the receiver's subviews as specified in the nib file from which they were unarchived or the programmatic interface for modifying the receiver's subview list. This ordering is also the reverse of the order in which hit-testing is done.

Note: The contents of this array may change at any time. If you intend to manipulate this array you should copy it rather than simply retain.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `superview` (page 3235)
- `addSubview:` (page 3139)
- `addSubview:positioned:relativeTo:` (page 3140)
- `removeFromSuperview` (page 3202)

Related Sample Code
AnimatingViews
EnhancedDataBurn
MenuItemView
MyPhoto
WhackedTV

Declared In
`NSView.h`

superview

Returns the receiver's superview, or `nil` if it has none.

```
- (NSView *)superview
```

Discussion
When applying this method iteratively or recursively, be sure to compare the returned view object to the content view of the window to avoid proceeding out of the view hierarchy.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `window` (page 3248)

- subviews (page 3235)
- removeFromSuperview (page 3202)

Related Sample Code
GLUT
MatrixMixerTest
Quartz Composer WWDC 2005 TextEdit
Reducer
TextSizingExample

Declared In
NSView.h

tag

Returns the receiver's tag, an integer that you can use to identify view objects in your application.

```
- (NSInteger)tag
```

Discussion
The default implementation returns –1. Subclasses can override this method to provide individual tags, possibly adding storage and a setTag: method (which NSView doesn't define).

Availability
Available in Mac OS X v10.0 and later.

See Also
- viewWithTag: (page 3245)

Related Sample Code
CIVideoDemoGL
FunHouse
PDF Annotation Editor
Quartz2DBasics
Sketch+Accessibility

Declared In
NSView.h

toolTip

Returns the text for the view's tool tip.

```
- (NSString *)toolTip
```

Return Value
The tool tip text or nil if the view doesn't currently display tool tip text

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setToolTip:` (page 3229)

Declared In
`NSView.h`

trackingAreas

Returns an array of the receiver's tracking areas.

```
- (NSArray *)trackingAreas
```

Return Value
An array of the receiver's tracking areas (instances of `NSTrackingArea`). If the receiver has no tracking areas, returns an empty array.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
MenuItemView
PhotoSearch

Declared In
`NSView.h`

translateOriginToPoint:

Translates the receiver's coordinate system so that its origin moves to a new location.

```
- (void)translateOriginToPoint:(NSPoint)newOrigin
```

Parameters
newOrigin
A point that specifies the new origin.

Discussion
In the process, the origin of the receiver's bounds rectangle is shifted by (*-newOrigin*.x, *-newOrigin*.y). This method neither redisplays the receiver nor marks it as needing display. You must do this yourself with `display` (page 3163) or `setNeedsDisplay:` (page 3225).

Note the difference between this method and setting the bounds origin. Translation effectively moves the image inside the bounds rectangle, while setting the bounds origin effectively moves the rectangle over the image. The two are in a sense inverse, although translation is cumulative, and setting the bounds origin is absolute.

This method posts an `NSViewBoundsDidChangeNotification` (page 3256) to the default notification center if the receiver is configured to do so.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setBoundsOrigin:` (page 3214)
- `setBounds:` (page 3214)
- `setPostsBoundsChangedNotifications:` (page 3226)

Related Sample Code
Polygons

Declared In
`NSView.h`

translateRectsNeedingDisplayInRect:by:

Translates the display rectangles by the specified delta.

```
- (void)translateRectsNeedingDisplayInRect:(NSRect)clipRect
    by:(NSSize)delta
```

Parameters

clipRect
A rectangle defining the region of the receiver, typically the receiver's bounds.

delta
A NSSize structure that specifies an offset from from aRect's origin.

Discussion
This method performs the shifting of dirty rectangles that an equivalent `scrollRect:by:` (page 3210) operation would cause, without performing the actual scroll operation. It is only useful in very rare cases where a view implements its own low-level scrolling mechanics.

This method:

1. Collects the receiving view's dirty rectangles.
2. Clears all dirty rectangles in the intersection of *clipRect* and the view's bounds.
3. Shifts the retrieved rectangles by the *delta* offset.
4. Clips the result to the intersection of *clipRect* and the view's bounds
5. Marks the resultant rectangles as needing display.

The developer must ensure that *clipRect* and *delta* are pixel-aligned in order to guarantee correct drawing. See "The View Coordinate System" in *View Programming Guide* for a description of how to pixel-align view coordinates.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

unlockFocus

Balances an earlier `lockFocus` (page 3187) or `lockFocusIfCanDraw` (page 3188) message; restoring the focus to the previously focused view is necessary.

```
- (void)unlockFocus
```

Discussion
Raises an `NSInvalidArgumentException` if invoked on the wrong view.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- allocateGState` (page 3146)

Related Sample Code
CocoaAUHost
Color Sampler
GLUT
QTKitFrameStepper
VideoViewer

Declared In
`NSView.h`

unregisterDraggedTypes

Unregisters the receiver as a possible destination in a dragging session.

```
- (void)unregisterDraggedTypes
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- registerForDraggedTypes:` (page 3201)

Declared In
`NSView.h`

updateTrackingAreas

Invoked automatically when the view's geometry changes such that its tracking areas need to be recalculated.

```
- (void)updateTrackingAreas
```

Discussion
You should override this method to remove out of date tracking areas and add recomputed tracking areas; your implementation should call `super`.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
PhotoSearch

Declared In
`NSView.h`

viewDidEndLiveResize

Informs the receiver of the end of a live resize.

```
- (void)viewDidEndLiveResize
```

Discussion
In the simple case, a view is sent `viewWillStartLiveResize` (page 3244) before the first resize operation on the containing window and `viewDidEndLiveResize` after the last resize operation. A view that is repeatedly added and removed from a window during live resize will receive only one `viewWillStartLiveResize` (on the first time it is added to the window) and one `viewDidEndLiveResize` (when the window has completed the live resize operation). This allows a superview such as `NSBrowser` object to add and remove its `NSMatrix` subviews during live resize without the `NSMatrix` receiving multiple calls to these methods.

A view might allocate data structures to cache-drawing information in `viewWillStartLiveResize` (page 3244) and should clean up these data structures in `viewDidEndLiveResize`. In addition, a view that does optimized drawing during live resize might want to do full drawing after `viewDidEndLiveResize`, although a view should not assume that it has a drawing context in `viewDidEndLiveResize` (since it may have been removed from the window during live resize). A view that wants to redraw itself after live resize should call `[self setNeedsDisplay:YES]` in `viewDidEndLiveResize`.

A view subclass should call `super` from these methods.

Availability
Available in Mac OS X v10.1 and later.

See Also
- `viewWillStartLiveResize` (page 3244)
- `inLiveResize` (page 3180)

Related Sample Code
GLUT
ImageApp
MyPhoto

Declared In
`NSView.h`

viewDidHide

Invoked when the receiver is hidden, either directly, or in response to an ancestor being hidden.

```
- (void)viewDidHide
```

Discussion
The receiver receives this message when its `isHiddenOrHasHiddenAncestor` (page 3182) state goes from `NO` to `YES`. This will happen when the view or an ancestor is marked as hidden, or when the view or an ancestor is inserted into a new view hierarchy.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSView.h`

viewDidMoveToSuperview

Informs the receiver that its superview has changed (possibly to `nil`).

```
- (void)viewDidMoveToSuperview
```

Discussion
The default implementation does nothing; subclasses can override this method to perform whatever actions are necessary.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `viewDidMoveToWindow` (page 3241)
- `viewWillMoveToSuperview:` (page 3243)
- `viewWillMoveToWindow:` (page 3243)

Declared In
`NSView.h`

viewDidMoveToWindow

Informs the receiver that it has been added to a new view hierarchy.

```
- (void)viewDidMoveToWindow
```

Discussion
The default implementation does nothing; subclasses can override this method to perform whatever actions are necessary.

`window` (page 3248) may return `nil` when this method is invoked, indicating that the receiver does not currently reside in any window. This occurs when the receiver has just been removed from its superview or when the receiver has just been added to a superview that does not itself have a window. Overrides of this method may choose to ignore such cases if they are not of interest.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `viewDidMoveToSuperview` (page 3241)
- `viewWillMoveToSuperview:` (page 3243)

- viewWillMoveToWindow: (page 3243)

Related Sample Code
ClockControl
DispatchFractal
iChatTheater
SpotlightAPI
WhackedTV

Declared In
NSView.h

viewDidUnhide

Invoked when the receiver is unhidden, either directly, or in response to an ancestor being unhidden

```
- (void)viewDidUnhide
```

Discussion
The receiver receives this message when its isHiddenOrHasHiddenAncestor state goes from YES to NO. This can happen when the view or an ancestor is marked as not hidden, or when the view or an ancestor is removed from its containing view hierarchy.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSView.h

viewWillDraw

Informs the receiver that it will be required to draw content.

```
- (void)viewWillDraw
```

Discussion
In response to receiving one of the display... messages the receiver will recurse down the view hierarchy, sending this message to each of the views that may be involved in the display operation.

Subclasses can override this method to move or resize views, mark additional areas as requiring display, or other actions that can best be deferred until they are required for drawing. During the recursion, sending of setNeedsDisplay: (page 3225) and setNeedsDisplayInRect: (page 3225) messages to views in the hierarchy that's about to be drawn is valid and supported, and will affect the assessment of the total area to be rendered in that drawing pass.

A subclass's implementation of viewWillDraw can use the existing NSView getRectsBeingDrawn:count: (page 3176) method to obtain a list of rectangles that bound the affected area, enabling it to restrict its efforts to that area.

The following is an example of a generic subclass implementation:

```
- (void)viewWillDraw {
```

```
    // Perform some operations before recursing for descendants.

    // Now recurse to handle all our descendants.
    // Overrides must call up to super like this.
    [super viewWillDraw];

    // Perform some operations that might depend on descendants
    //  already having had a chance to update.
}
```

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
AnimatedTableView

Declared In
`NSView.h`

viewWillMoveToSuperview:

Informs the receiver that its superview is about to change to the specified superview (which may be `nil`).

```
- (void)viewWillMoveToSuperview:(NSView *)newSuperview
```

Parameters

newSuperview
A view object that will be the new superview of the receiver.

Discussion
Subclasses can override this method to perform whatever actions are necessary.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `viewDidMoveToSuperview` (page 3241)
- `viewDidMoveToWindow` (page 3241)
- `viewWillMoveToWindow:` (page 3243)

Declared In
`NSView.h`

viewWillMoveToWindow:

Informs the receiver that it's being added to the view hierarchy of the specified window object (which may be `nil`).

```
- (void)viewWillMoveToWindow:(NSWindow *)newWindow
```

Parameters

newWindow
A window object that will be at the root of the receiver's new view hierarchy.

Discussion
Subclasses can override this method to perform whatever actions are necessary.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `viewDidMoveToSuperview` (page 3241)
- `viewDidMoveToWindow` (page 3241)
- `viewWillMoveToSuperview:` (page 3243)

Declared In
`NSView.h`

viewWillStartLiveResize

Informs the receiver of the start of a live resize.

```
- (void)viewWillStartLiveResize
```

Discussion
In the simple case, a view is sent `viewWillStartLiveResize` before the first resize operation on the containing window and `viewDidEndLiveResize` (page 3240) after the last resize operation. A view that is repeatedly added and removed from a window during live resize will receive only one `viewWillStartLiveResize` (on the first time it is added to the window) and one `viewDidEndLiveResize` (when the window has completed the live resize operation). This allows a superview such as `NSBrowser` object to add and remove its `NSMatrix` subviews during live resize without the `NSMatrix` object receiving multiple calls to these methods.

A view might allocate data structures to cache-drawing information in `viewWillStartLiveResize` and should clean up these data structures in `viewDidEndLiveResize` (page 3240). In addition, a view that does optimized drawing during live resize might want to do full drawing after `viewDidEndLiveResize`, although a view should not assume that it has a drawing context in `viewDidEndLiveResize` (since it may have been removed from the window during live resize). A view that wants to redraw itself after live resize should call `[self setNeedsDisplay:YES]` in `viewDidEndLiveResize`.

A view subclass should call `super` from these methods.

Availability
Available in Mac OS X v10.1 and later.

See Also
- `viewDidEndLiveResize` (page 3240)
- `inLiveResize` (page 3180)

Related Sample Code
GLUT
MyPhoto

Declared In
`NSView.h`

viewWithTag:

Returns the receiver's nearest descendant (including itself) with a specific tag, or `nil` if no subview has that tag.

```
- (id)viewWithTag:(NSInteger)aTag
```

Parameters

aTag

An integer identifier associated with a view object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- tag` (page 3236)

Related Sample Code
AnimatedTableView
EnhancedDataBurn
PrefsPane

Declared In
`NSView.h`

visibleRect

Returns the portion of the receiver not clipped by its superviews.

```
- (NSRect)visibleRect
```

Return Value
A rectangle defining the unclipped portion of the receiver.

Discussion
Visibility for this method is defined quite simply and doesn't account for whether other `NSView` objects (or windows) overlap the receiver or whether the receiver has a window at all. This method returns `NSZeroRect` if the receiver is effectively hidden.

During a printing operation the visible rectangle is further clipped to the page being imaged.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setHidden:` (page 3222)
`isVisible` (page 3342) (NSWindow)
`documentVisibleRect` (page 2346) (NSScrollView)
`documentVisibleRect` (page 634) (NSClipView)

Related Sample Code
CIAnnotation
ImageKitDemo
Quartz Composer WWDC 2005 TextEdit

Rulers
TextLinks

Declared In
NSView.h

wantsDefaultClipping

Returns whether the Application Kit's default clipping provided to `drawRect:` (page 3170) implementations is in effect.

```
- (BOOL)wantsDefaultClipping
```

Return Value
`YES` if the default clipping is in effect, `NO` otherwise. By default, this method returns `YES`.

Discussion
Subclasses may override this method to return `NO` if they want to suppress the default clipping. They may want to do this in situations where drawing performance is critical to avoid the cost of setting up, enforcing, and cleaning up the clip path

A view that overrides this method to refuse the default clipping must either set up whatever clipping it requires or constrain its drawing exactly to the list of rectangles returned by `getRectsBeingDrawn:count:` (page 3176). Failing to do so could result in corruption of other drawing in the view's window.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
VideoViewer

Declared In
NSView.h

wantsLayer

Returns a Boolean value that indicates whether the receiver is using a layer as its backing store.

```
- (BOOL)wantsLayer
```

Return Value
`YES` if the receiver is using a Core Animation layer as its backing store, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSView.h

wantsRestingTouches

Returns whether the view wants resting touches.

```
- (BOOL)wantsRestingTouches
```

Return Value
`YES` if the view wants resting touches, otherwise `NO`. The default is `NO`.

Discussion
A resting touch occurs when a user rests their thumb on a device (for example, the glass trackpad of a MacBook).

By default, these touches are not delivered and are not included in the event's set of touches. Touches may transition in and out of resting at any time. Unless the view wants restingTouches, began / ended events are simulated as touches transition from resting to active and vice versa.

In general resting touches should be ignored.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setWantsRestingTouches:` (page 3230)

Declared In
`NSView.h`

widthAdjustLimit

Returns the fraction (from 0.0 to 1.0) of the page that can be pushed onto the next page during automatic pagination to prevent items such as small images or text columns from being divided across pages.

```
- (CGFloat)widthAdjustLimit
```

Discussion
This fraction is used to calculate the right edge limit for a `adjustPageWidthNew:left:right:limit:` (page 3144) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- heightAdjustLimit` (page 3178)

Declared In
`NSView.h`

willRemoveSubview:

Overridden by subclasses to perform additional actions before subviews are removed from the receiver.

```
- (void)willRemoveSubview:(NSView *)subview
```

Parameters

subview

The subview that will be removed.

Discussion

This method is invoked when *subview* receives a `removeFromSuperview` (page 3202) message or *subview* is removed from the receiver due to it being added to another view with `addSubview:` (page 3139).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSView.h`

window

Returns the receiver's window object, or `nil` if it has none.

```
- (NSWindow *)window
```

Availability

Available in Mac OS X v10.0 and later.

See Also

`- superview` (page 3235)

Related Sample Code

CIAnnotation

CIVideoDemoGL

GLUT

Quartz Composer WWDC 2005 TextEdit

Sketch-112

Declared In

`NSView.h`

writeEPSInsideRect:toPasteboard:

Writes EPS data that draws the region of the receiver within a specified rectangle onto a pasteboard.

```
- (void)writeEPSInsideRect:(NSRect)aRect
    toPasteboard:(NSPasteboard *)pboard
```

Parameters

aRect

A rectangle defining the region.

pboard

An object representing a pasteboard.

Availability

Available in Mac OS X v10.0 and later.

See Also
- dataWithEPSInsideRect: (page 3162)

Declared In
NSView.h

writePDFInsideRect:toPasteboard:

Writes PDF data that draws the region of the receiver within a specified rectangle onto a pasteboard.

```
- (void)writePDFInsideRect:(NSRect)aRect
    toPasteboard:(NSPasteboard *)pboard
```

Parameters

aRect
A rectangle defining the region.

pboard
An object representing a pasteboard.

Availability
Available in Mac OS X v10.0 and later.

See Also
- dataWithPDFInsideRect: (page 3162)

Declared In
NSView.h

Constants

NSBorderType

These constants specify the type of a view's border.

```
enum {
    NSNoBorder     = 0,
    NSLineBorder   = 1,
    NSBezelBorder  = 2,
    NSGrooveBorder = 3
};
typedef NSUInteger NSBorderType;
```

Constants
NSBezelBorder
A concave border that makes the view look sunken.

Available in Mac OS X v10.0 and later.

Declared in NSView.h.

`NSGrooveBorder`
A thin border that looks etched around the image.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

`NSLineBorder`
A black line border around the view.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

`NSNoBorder`
No border.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

Declared In
`NSView.h`

Resizing masks

These constants are used by `setAutoresizingMask:` (page 3212).

```
enum {
   NSViewNotSizable     = 0,
   NSViewMinXMargin     = 1,
   NSViewWidthSizable   = 2,
   NSViewMaxXMargin     = 4,
   NSViewMinYMargin     = 8,
   NSViewHeightSizable  = 16,
   NSViewMaxYMargin     = 32
};
```

Constants
`NSViewNotSizable`
The receiver cannot be resized.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

`NSViewMinXMargin`
The left margin between the receiver and its superview is flexible.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

`NSViewWidthSizable`
The receiver's width is flexible.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

`NSViewMaxXMargin`
The right margin between the receiver and its superview is flexible.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

`NSViewMinYMargin`
The bottom margin between the receiver and its superview is flexible.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

`NSViewHeightSizable`
The receiver's height is flexible.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

`NSViewMaxYMargin`
The top margin between the receiver and its superview is flexible.

Available in Mac OS X v10.0 and later.

Declared in `NSView.h`.

Declared In
`NSView.h`

NSToolTipTag

This type describes the rectangle used to identify a tool tip rectangle.

```
typedef NSInteger NSToolTipTag;
```

Discussion
See the methods `addToolTipRect:owner:userData:` (page 3141) and `removeToolTip:` (page 3203).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

NSTrackingRectTag

This type describes the rectangle used to track the mouse.

```
typedef NSInteger NSTrackingRectTag;
```

Discussion
See the methods `addTrackingRect:owner:userData:assumeInside:` (page 3142) and `removeTrackingRect:` (page 3204).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

Full Screen Mode Options

These constants are keys that you can use in the options dictionary in `enterFullScreenMode:withOptions:` (page 3173) and `exitFullScreenModeWithOptions:` (page 3174).

```
NSString * const NSFullScreenModeAllScreens;
NSString * const NSFullScreenModeSetting;
NSString * const NSFullScreenModeWindowLevel;
NSString * const NSFullScreenModeApplicationPresentationOptions;
```

Constants

`NSFullScreenModeAllScreens`

Key whose corresponding value specifies whether the view should take over all screens.

The corresponding value is an instance of `NSNumber` containing a Boolean value.

Available in Mac OS X v10.5 and later.

Declared in `NSView.h`.

`NSFullScreenModeSetting`

Key whose corresponding value specifies the the full screen mode setting.

The corresponding value is an instance of `NSDictionary` that contains keys specified in `Display Mode Standard Properties` and `Display Mode Optional Properties` in *Quartz Display Services Reference*.

When the `NSFullScreenModeApplicationPresentationOptions` (page 3252) is specified in the options dictionary specifying this option as well will cause an exception.

Available in Mac OS X v10.5 and later.

Declared in `NSView.h`.

`NSFullScreenModeWindowLevel`

Key whose corresponding value specifies the screen mode window level.

The corresponding value is an instance of `NSNumber` containing an integer value.

Available in Mac OS X v10.5 and later.

Declared in `NSView.h`.

`NSFullScreenModeApplicationPresentationOptions`

Key whose corresponding value specifies the application presentation options..

The corresponding value is an instance of `NSNumber` containing an unsigned integer value of `NSApplicationPresentationOptions` (page 187). Those options can be combined using the C bit-wise `OR` operator before created the `NSNumber` instance. See *NSApplication Class Reference* constants section `NSApplicationPresentationOptions` (page 187) for more information on these options.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

Declared In

`NSView.h`

NSViewLayerContentsRedrawPolicy

These constants specify how layer resizing is handled when a view is layer-backed or layer-hosting. See `layerContentsRedrawPolicy` (page 3186) and `setLayerContentsRedrawPolicy:` (page 3224) for more information.

```
enum {
    NSViewLayerContentsRedrawNever                  = 0,
    NSViewLayerContentsRedrawOnSetNeedsDisplay      = 1,
    NSViewLayerContentsRedrawDuringViewResize       = 2,
    NSViewLayerContentsRedrawBeforeViewResize       = 3
};
typedef NSInteger NSViewLayerContentsRedrawPolicy;
```

Constants

`NSViewLayerContentsRedrawNever`

Leave the layer's contents alone. Never mark the layer as needing display, or draw the view's contents to the layer. This is how developer created layers (layer-hosting views) are treated.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsRedrawOnSetNeedsDisplay`

Any of the `setNeedsDisplay...` methods sent to the view will cause the view redraw the affected layer parts by invoking the view's `drawRect:` (page 3170), but neither the layer or the view are marked as needing display when the view's size changes.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsRedrawDuringViewResize`

Resize the view's backing-layer and redraw the view to the layer when the view's size changes. If the resize is animated, AppKit will drive the resize animation itself and will do this resize and redraw at each step of the animation. Affected parts of the layer will also be redrawn when the view is marked as needing display. This mode is a superset of `NSViewLayerContentsRedrawOnSetNeedsDisplay` (page 3253). This is the way that layer-backed views are currently treated.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsRedrawBeforeViewResize`

Resize the layer and redraw the view to the layer when the view's size changes. This will be done just once at the beginning of a resize animation, not at each frame of the animation. Affected parts of the layer will also be redrawn when the view is marked as needing display. This mode is a superset of `NSViewLayerContentsRedrawOnSetNeedsDisplay` (page 3253).

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

NSViewLayerContentsPlacement

These constants specify the location of the layer content when the content is not re-rendered in response to view resizing. See `setLayerContentsPlacement:` (page 3223) for more information.

```
enum {
    NSViewLayerContentsPlacementScaleAxesIndependently      =  0,
    NSViewLayerContentsPlacementScaleProportionallyToFit    =  1,
    NSViewLayerContentsPlacementScaleProportionallyToFill   =  2,
    NSViewLayerContentsPlacementCenter                      =  3,
    NSViewLayerContentsPlacementTop                         =  4,
    NSViewLayerContentsPlacementTopRight                    =  5,
    NSViewLayerContentsPlacementRight                       =  6,
    NSViewLayerContentsPlacementBottomRight                 =  7,
    NSViewLayerContentsPlacementBottom                      =  8,
    NSViewLayerContentsPlacementBottomLeft                  =  9,
    NSViewLayerContentsPlacementLeft                        = 10,
    NSViewLayerContentsPlacementTopLeft                     = 11
};
typedef NSInteger NSViewLayerContentsPlacement;
```

Constants

`NSViewLayerContentsPlacementScaleAxesIndependently`

The content is resized to fit the entire bounds rectangle.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsPlacementScaleProportionallyToFit`

The content is resized to fit the bounds rectangle, preserving the aspect of the content. If the content does not completely fill the bounds rectangle, the content is centered in the partial axis.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsPlacementScaleProportionallyToFill`

The content is resized to completely fill the bounds rectangle, while still preserving the aspect of the content. The content is centered in the axis it exceeds.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsPlacementCenter`

The content is horizontally and vertically centered in the bounds rectangle.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsPlacementTop`

The content is horizontally centered at the top-edge of the bounds rectangle.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsPlacementTopRight`

The content is positioned in the top-right corner of the bounds rectangle.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsPlacementRight`

The content is vertically centered at the right-edge of the bounds rectangle.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsPlacementBottomRight`
The content is positioned in the bottom-right corner of the bounds rectangle.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsPlacementBottom`
The content is horizontally centered at the bottom-edge of the bounds rectangle.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsPlacementBottomLeft`
The content is positioned in the bottom-left corner of the bounds rectangle.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsPlacementLeft`
The content is vertically centered at the left-edge of the bounds rectangle.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSViewLayerContentsPlacementTopLeft`
The content is positioned in the top-left corner of the bounds rectangle.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

NSDefinition Presentation Constants

These constants are used to control how definition windows are displayed by `showDefinitionForAttributedString:range:options:baselineOriginProvider:` (page 3233). If this option is unspecified, the definition will be shown in either of those presentation forms depending on the 'Contextual Menu:' setting in Dictionary application preferences.

```
NSString * const NSDefinitionPresentationTypeKey;
NSString * const NSDefinitionPresentationTypeOverlay;
NSString * const NSDefinitionPresentationTypeDictionaryApplication;
```

Constants

`NSDefinitionPresentationTypeKey`
An optional key in the options dictionary that specifies the presentation type of the definition display. It can have a value of `NSDefinitionPresentationTypeOverlay` (page 3255) or `NSDefinitionPresentationTypeDictionaryApplication` (page 3256).

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSDefinitionPresentationTypeOverlay`
A possible value of the `NSDefinitionPresentationTypeKey` (page 3255) dictionary key that produces a small overlay window at the string location,

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

`NSDefinitionPresentationTypeDictionaryApplication`
A possible value of the `NSDefinitionPresentationTypeKey` (page 3255) dictionary key that invokes Dictionary application to display the definition.

Available in Mac OS X v10.6 and later.

Declared in `NSView.h`.

Notifications

NSViewBoundsDidChangeNotification

Posted whenever the `NSView`'s bounds rectangle changes independently of the frame rectangle, if the `NSView` is configured using `setPostsBoundsChangedNotifications:` (page 3226) to post such notifications.

The notification object is the `NSView` object whose bounds rectangle has changed. This notification does not contain a *`userInfo`* dictionary.

The following methods can result in notification posting:

`setBounds:` (page 3214)
`setBoundsOrigin:` (page 3214)
`setBoundsRotation:` (page 3215)
`setBoundsSize:` (page 3216)
`translateOriginToPoint:` (page 3237)
`scaleUnitSquareToSize:` (page 3208)
`rotateByAngle:` (page 3208)

Note that the bounds rectangle resizes automatically to track the frame rectangle. Because the primary change is that of the frame rectangle, however, `setFrame:` (page 3218) and `setFrameSize:` (page 3221) don't result in a bounds-changed notification.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSView.h`

NSViewFocusDidChangeNotification

Deprecated notification that was posted for an `NSView` object and each of its descendants (recursively) whenever the frame or bounds geometry of the view changed.

In Mac OS X v10.4 and later, this notification is no longer posted. In earlier version of Mac OS X, use `NSViewBoundsDidChangeNotification` and `NSViewFrameDidChangeNotification` instead to get the same information provided by this notification.

The notification object is the view whose geometry changed. This notification does not contain a *`userInfo`* dictionary.

Availability
Deprecated in Mac OS X v10.4 and later.

See Also
NSViewBoundsDidChangeNotification (page 3256)
NSViewFrameDidChangeNotification (page 3257)

Declared In
NSView.h

NSViewFrameDidChangeNotification

Posted whenever the view's frame rectangle changes, if the view is configured using setPostsFrameChangedNotifications: (page 3227) to post such notifications.

The notification object is the NSView object whose frame rectangle has changed. This notification does not contain a *userInfo* dictionary.

The following methods can result in notification posting:

- setFrame: (page 3218)
- setFrameOrigin: (page 3220)
- setFrameRotation: (page 3221)
- setFrameSize: (page 3221)

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSView.h

NSViewDidUpdateTrackingAreasNotification

Posted whenever an NSView object recalculates its tracking areas. It is sent after the view receives updateTrackingAreas (page 3239).

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSView.h

NSViewGlobalFrameDidChangeNotification

Posted whenever an NSView object that has attached surfaces (that is, NSOpenGLContext objects) moves to a different screen, or other cases where the NSOpenGLContext object needs to be updated. The notification object is the surface's view. This notification does not contain a *userInfo* dictionary.

Availability
Available in Mac OS X v10.1 and later.

Declared In
NSView.h

NSViewAnimation Class Reference

Inherits from	NSAnimation : NSObject
Conforms to	NSCoding (NSAnimation) NSCopying (NSAnimation) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSAnimation.h
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Cocoa Drawing Guide
Related sample code	From A View to A Movie iSpend QTCoreVideo301 Reducer

Overview

The `NSViewAnimation` class, a public subclass of `NSAnimation`, offers a convenient way to animate multiple views and windows. The animation effects you can achieve are limited to changes in frame location and size, and to fade-in and fade-out effects.

An `NSViewAnimation` object takes an array of dictionaries from which it determines the objects to animate and the effects to apply to them. Each dictionary must have a target object and, optionally, properties that specify beginning and ending frame and whether to fade in or fade out. (See "View Animation Dictionary Keys" (page 3261) for further information.) Animations with `NSViewAnimation` are, by default, in non-blocking mode over a duration of 0.5 seconds using the ease in-out animation curve. But you can configure the animation to have any duration, curve, frame rate, and blocking mode. You may also set progress marks, assign a delegate, and implement delegation methods in order to animate view and windows concurrent with the ones specified as targets in the view-animation dictionary.

Invoking the `NSAnimation` `stopAnimation` (page 112) method on a running `NSViewAnimation` object moves the animation to the end frame.

Tasks

Initializing an NSViewAnimation Object

- `initWithViewAnimations:` (page 3260)
  Returns an `NSViewAnimation` object initialized with the supplied information.

Getting and Setting View-animation Dictionaries

- `setViewAnimations:` (page 3261)
  Sets the dictionaries defining the objects to animate.
- `viewAnimations` (page 3261)
  Returns the array of dictionaries defining the objects to animate.

Instance Methods

initWithViewAnimations:

Returns an `NSViewAnimation` object initialized with the supplied information.

```
- (id)initWithViewAnimations:(NSArray *)viewAnimations
```

Parameters

viewAnimations

An array of `NSDictionary` objects. Each dictionary specifies a view or window to animate and the effect to apply. *viewAnimations* can be `nil`, but you must later set the required array of dictionaries with `setViewAnimations:` (page 3261) if you want to use the capabilities of the `NSViewAnimation` class. See"View Animation Dictionary Keys" (page 3261) for a description of valid keys and values for dictionaries in *viewAnimations*.

Return Value

The created `NSViewAnimation` object or `nil` if there was a problem initializing the object.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

From A View to A Movie

iSpend

QTCoreVideo301

Reducer

Declared In

`NSAnimation.h`

setViewAnimations:

Sets the dictionaries defining the objects to animate.

```
- (void)setViewAnimations:(NSArray *)viewAnimations
```

Parameters

viewAnimations

An array of `NSDictionary` objects. Each dictionary specifies a view or window to animate and the effect to apply. Pass in `nil` to remove the current list of dictionaries. See "View Animation Dictionary Keys" (page 3261) for a description of valid keys and values for dictionaries in *viewAnimations*.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `viewAnimations` (page 3261)

Related Sample Code
From A View to A Movie

Declared In
`NSAnimation.h`

viewAnimations

Returns the array of dictionaries defining the objects to animate.

```
- (NSArray *)viewAnimations
```

Discussion
Each dictionary in the returned array specifies a view or window to animate and the effect to apply.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setViewAnimations:` (page 3261)

Declared In
`NSAnimation.h`

Constants

View Animation Dictionary Keys

The following string constants are keys for the dictionaries in the array passed into `initWithViewAnimations:` (page 3260) and `setViewAnimations:` (page 3261).

```
NSString *NSViewAnimationTargetKey;
NSString *NSViewAnimationStartFrameKey;
NSString *NSViewAnimationEndFrameKey;
NSString *NSViewAnimationEffectKey;
```

Constants

`NSViewAnimationTargetKey`

The target of the animation.

The target can be either an `NSView` object or an `NSWindow` object. This property is required.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

`NSViewAnimationStartFrameKey`

The size and location of the window or view at the start of the animation.

The size and location are specified by an `NSRect` structure encoded in an `NSValue` object. This property is optional. If it is not specified, `NSViewAnimation` uses the frame of the window or view at the start of the animation.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

`NSViewAnimationEndFrameKey`

The size and location of the window or view at the end of the animation.

The size and location are specified by an `NSRect` structure encoded in an `NSValue` object. This property is optional. If it is not specified, `NSViewAnimation` uses the frame of the window or view at the start of the animation. If the target is a view and the end frame is empty, the view is hidden at the end.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

`NSViewAnimationEffectKey`

An effect to apply to the animation.

Takes a string constant specifying fade-in or fade-out effects for the target: `NSViewAnimationFadeInEffect` and `NSViewAnimationFadeOutEffect`. If the target is a view and the effect is to fade out, the view is hidden at the end. If the effect is to fade in an initially hidden view and the end frame is non-empty, the view is unhidden at the end. If the target is a window, the window is ordered in or out as appropriate to the effect. This property is optional.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

Declared In

`NSAnimation.h`

Values for NSViewAnimationEffectKey

The following constants specify the animation effect to apply and are used as values for the animation effect property of the animation view. See the description of `NSViewAnimationEffectKey` for usage details.

```
NSString *NSViewAnimationFadeInEffect;
NSString *NSViewAnimationFadeOutEffect;
```

Constants

`NSViewAnimationFadeInEffect`

Specifies a fade-in type of effect.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

`NSViewAnimationFadeOutEffect`

Specifies a fade-out type of effect.

Available in Mac OS X v10.4 and later.

Declared in `NSAnimation.h`.

Availability

Available in Mac OS X v10.4 and later

Declared In

`NSAnimation.h`

NSViewController Class Reference

Inherits from	NSResponder : NSObject
Conforms to	NSCoding NSCoding (NSResponder) NSObject (NSObject) NSEditor (Informal Protocol) NSEditorRegistration (Informal Protocol)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSViewController.h
Related sample code	AnimatedTableView ComplexBrowser SourceView ViewController ZipBrowser

Overview

An `NSViewController` object manages a view, typically loaded from a nib file.

This management includes:

- Memory management of top-level objects similar to that of the `NSWindowController` class, taking the same care to prevent reference cycles when controls are bound to the nib file's owner that `NSWindowController` began taking in Mac OS v 10.4.
- Declaring a generic `representedObject` (page 3271) property, to make it easy to establish bindings in the nib to an object that isn't yet known at nib-loading time or readily available to the code that's doing the nib loading.
- Implementing the key-value binding `NSEditor` informal protocol, so that applications using `NSViewController` can easily make bound controls in the views commit or discard the changes the user is making.

`NSViewController` is meant to be highly reusable. For example, the `NSPageLayout` and `NSPrintPanel` `addAccessoryController:` (page 1854) methods take an `NSViewController` instance as the argument, and set the `representedObject` (page 3271) to the `NSPrintInfo` that is to be shown to the user. This allows a developer to easily create new printing accessory views using bindings and the `NSPrintInfo` class' new key-value coding and key-value observing compliance. When the user dismisses a printing panel,

`NSPageLayout` and `NSPrintPanel` each send `NSEditor` messages to each accessory view controller to ensure that the user's changes have been committed or discarded properly. The titles of the accessories are retrieved from the view controllers and shown to the user in pulldown menus that the user can choose from. If your application needs to dynamically associate relatively complex views with the objects that they present to the user, you might be able to reuse `NSViewController` in similar ways.

Tasks

Creating A View Controller

- `initWithNibName:bundle:` (page 3269)
  Returns an `NSViewController` object initialized to the nib file in the specified bundle.
- `loadView` (page 3269)
  Instantiate the receiver's view and set it.

Represented Object

- `setRepresentedObject:` (page 3271)
  Sets the object whose value is being presented in the receiver's view.
- `representedObject` (page 3271)
  Returns the object whose value is being presented in the receiver's view.

Nib Properties

- `nibBundle` (page 3270)
  Return the name of the nib bundle to be loaded to instantiate the receivers view.
- `nibName` (page 3270)
  Return the name of the nib to be loaded to instantiate the receivers view

View Properties

- `view` (page 3273)
  Returns the receiver's view.
- `setView:` (page 3272)
  Sets the receivers view to the specified object.
- `title` (page 3272)
  Returns the localized title of the receiver's view.
- `setTitle:` (page 3271)
  Sets the localized title of the receiver's view to the specified string.

NSEditor Conformance

- `commitEditingWithDelegate:didCommitSelector:contextInfo:` (page 3267)
  Attempt to commit any currently edited results of the receiver.
- `commitEditing` (page 3267)
  Returns whether the receiver was able to commit any pending edits.
- `discardEditing` (page 3268)
  Causes the receiver to discard any changes, restoring the previous values.

Instance Methods

commitEditing

Returns whether the receiver was able to commit any pending edits.

```
- (BOOL)commitEditing
```

Return Value
Returns `YES` if the changes were successfully applied to the model, `NO` otherwise.

Discussion
A commit is denied if the receiver fails to apply the changes to the model object, perhaps due to a validation error.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `commitEditingWithDelegate:didCommitSelector:contextInfo:` (page 3267)
- `discardEditing` (page 3268)

Declared In
`NSViewController.h`

commitEditingWithDelegate:didCommitSelector:contextInfo:

Attempt to commit any currently edited results of the receiver.

```
- (void)commitEditingWithDelegate:(id)delegate
    didCommitSelector:(SEL)didCommitSelector contextInfo:(void *)contextInfo
```

Parameters
delegate
An object that can serve as the receiver's delegate. It should implement the method specified by *didCommitSelector*.

didCommitSelector
A selector that is invoked on delegate.

contextInfo

Contextual information that is sent as the `contextInfo` argument to delegate when *didCommitSelector* is invoked.

Discussion
The receiver must have been registered as the editor of an object using `objectDidBeginEditing:` (page 3682), and has not yet been unregistered by a subsequent invocation of `objectDidEndEditing:` (page 3682). When the committing has either succeeded or failed, send the *delegate* the message specified by *didCommitSelector*.

The *didCommitSelector* method must have the following method signature:.

```
- (void)editor:(id)editor didCommit:(BOOL)didCommit contextInfo:(void
*)contextInfo
```

If an error occurs while attempting to commit, for example if key-value coding validation fails, an implementation of this method should typically send the receiver's view a`presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 2187) message, specifying the view's containing window.

You may find this method useful in some situations when you want to ensure that pending changes are applied before a change in user interface state. For example, you may need to ensure that changes pending in a text field are applied before a window is closed. See also `commitEditing` (page 3267) which performs a similar function but which allows you to handle any errors directly, although it provides no information beyond simple success/failure.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `commitEditing` (page 3267)
- `discardEditing` (page 3268)

Declared In
`NSViewController.h`

discardEditing

Causes the receiver to discard any changes, restoring the previous values.

```
- (void)discardEditing
```

Availability
Available in Mac OS X v10.5 and later.

See Also
- `commitEditing` (page 3267)

Declared In
`NSViewController.h`

initWithNibName:bundle:

Returns an `NSViewController` object initialized to the nib file in the specified bundle.

```
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
```

Parameters

nibNameOrNil

The name of the nib file, without any leading path information.

nibBundleOrNil

The bundle in which to search for the nib file. If you specify nil, this method looks for the nib file in the main bundle.

Return Value

The initialized `NSViewController` object or `nil` if there were errors during initialization or the nib file could not be located.

Discussion

The `NSViewController` object looks for the nib file in the bundle's language-specific project directories first, followed by the Resources directory.

The specified nib should typically have the class of the file's owner set to `NSViewController`, or a custom subclass, with the `view` outlet connected to a view.

If you pass in a `nil` for *nibNameOrNil* then `nibName` (page 3270) will return `nil` and `loadView` (page 3269) will throw an exception; in this case you must invoke `setView:` (page 3272) before `view` (page 3273) is invoked, or override `loadView` (page 3269).

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

AnimatedTableView

ComplexBrowser

SourceView

ViewController

ZipBrowser

Declared In

`NSViewController.h`

loadView

Instantiate the receiver's view and set it.

```
- (void)loadView
```

Discussion

The default implementation of this method invokes `nibName` (page 3270) and `nibBundle` (page 3270) and then uses the `NSNib` class to load the nib with the receiver as the file's owner. If the `view` outlet of the file's owner in the nib is properly connected, the regular nib loading machinery will send the receiver a `setView:` (page 3272) message.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
AnimatedTableView

Declared In
NSViewController.h

nibBundle

Return the name of the nib bundle to be loaded to instantiate the receivers view.

```
- (NSBundle *)nibBundle
```

Return Value
The name of the nib bundle.

Discussion
The default implementation returns whatever value was passed to the initializer.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `initWithNibName:bundle:` (page 3269)
- `nibName` (page 3270)

Declared In
NSViewController.h

nibName

Return the name of the nib to be loaded to instantiate the receivers view

```
- (NSString *)nibName
```

Return Value
The name of the nib.

Discussion
The default implementation returns whatever value was passed to the initializer.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `initWithNibName:bundle:` (page 3269)
- `nibBundle` (page 3270)

Declared In
NSViewController.h

representedObject

Returns the object whose value is being presented in the receiver's view.

```
- (id)representedObject
```

Return Value
The object whose value is presented in the receiver's view.

Discussion
This class is key-value coding and key-value observing compliant for `representedObject` (page 3271) so when you use it as the file's owner of a view's nib you can bind controls to the file's owner using key paths that start with `representedObject`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setRepresentedObject:` (page 3271)

Declared In
`NSViewController.h`

setRepresentedObject:

Sets the object whose value is being presented in the receiver's view.

```
- (void)setRepresentedObject:(id)representedObject
```

Parameters
representedObject
The object whose value is presented in the receiver's view.

Discussion
The default implementation of this method doesn't copy the passed-in object, it retains it. In another words, *representedObject* is a to-one relationship, not an attribute.

This class is key-value coding and key-value observing compliant for `representedObject` (page 3271) so when you use it as the file's owner of a view's nib you can bind controls to the file's owner using key paths that start with `representedObject`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- representedObject` (page 3271)

Declared In
`NSViewController.h`

setTitle:

Sets the localized title of the receiver's view to the specified string.

```
- (void)setTitle:(NSString *)title
```

Parameters

title

The localized title of the receiver view.

Discussion

`NSViewController` does not use the title property directly. This property is here because so many anticipated uses of this class will involve letting the user choose among multiple named views using a pulldown menu or some other user interface.

This class is key value coding and key value compliant for this property.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- title` (page 3272)

Declared In

`NSViewController.h`

setView:

Sets the receivers view to the specified object.

```
- (void)setView:(NSView *)view
```

Parameters

view

The view the receiver should manage.

Discussion

You can invoke this method immediately after creating the object to specify a view that's created in a different manner than the receiver's default implementation would provide.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- view` (page 3273)

Declared In

`NSViewController.h`

title

Returns the localized title of the receiver's view.

```
- (NSString *)title
```

Return Value

The localized title of the receiver's view

Discussion
`NSViewController` does not use the title property directly. This property is here because so many anticipated uses of this class will involve letting the user choose among multiple named views using a pulldown menu or some other user interface.

This class is key value coding and key value compliant for this property.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setTitle:` (page 3271)

Declared In
`NSViewController.h`

view

Returns the receiver's view.

```
- (NSView *)view
```

Return Value
The receiver's view object.

Discussion
The default implementation of this method first invokes `loadView` (page 3269) if the view hasn't been set yet.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setView:` (page 3272)
`- loadView` (page 3269)

Related Sample Code
AnimatedTableView
ViewController

Declared In
`NSViewController.h`

NSWindow Class Reference

Inherits from	NSResponder : NSObject
Conforms to	NSUserInterfaceValidations NSAnimatablePropertyContainer NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSWindow.h AppKit/NSDrawer.h AppKit/NSGraphics.h
Companion guide	Window Programming Guide
Related sample code	Cocoa Tips and Tricks GLUT ImageClient MyPhoto Quartz Composer WWDC 2005 TextEdit

Overview

The `NSWindow` class defines objects (known as **windows**) that manage and coordinate the windows an application displays on the screen. A single `NSWindow` object corresponds to at most one onscreen window. The two principal functions of a window are to provide an area in which views can be placed and to accept and distribute, to the appropriate views, events the user instigates through actions with the mouse and keyboard.

Note: Although the `NSWindow` class inherits the `NSCoding` protocol from `NSResponder`, the class does not support coding. Legacy support for archivers exists but its use is deprecated and may not work. Any attempt to archive or unarchive an `NSWindow` object using a keyed coding object raises an `NSInvalidArgumentException` exception.

Tasks

Creating Windows

- `initWithContentRect:styleMask:backing:defer:` (page 3332)
  Initializes the window with the specified values.
- `initWithContentRect:styleMask:backing:defer:screen:` (page 3333)
  Initializes an allocated window with the specified values.

Configuring Windows

- `styleMask` (page 3402)
  Returns the window's style mask, indicating what kinds of control items it displays.
- `setStyleMask:` (page 3398)
  Sets the window's style mask to the given value.
- `worksWhenModal` (page 3409)
  Indicates whether the window is able to receive keyboard and mouse events even when some other window is being run modally.
- `alphaValue` (page 3298)
  Returns the window's alpha value.
- `setAlphaValue:` (page 3367)
  Applies a given alpha value to the entire window.
- `backgroundColor` (page 3301)
  Returns the color of the window's background.
- `setBackgroundColor:` (page 3370)
  Sets the window's background color to the given color.
- `colorSpace` (page 3309)
  Returns the window's color space.
- `setColorSpace:` (page 3373)
  Sets the window's color space.
- `contentView` (page 3313)
  Returns the window's content view, the highest accessible `NSView` object in the window's view hierarchy.
- `setContentView:` (page 3376)
  Makes a given view the window's content view.

- `canHide` (page 3306)
  Indicates whether the window can be hidden when its application becomes hidden (during execution of the `NSApplication hide:` (page 148) method).
- `setCanHide:` (page 3372)
  Specifies whether the window can be hidden when its application becomes hidden (during execution of the `NSApplication hide:` (page 148) method).
- `isOnActiveSpace` (page 3340)
  Indicates whether the window is on the currently active space.
- `hidesOnDeactivate` (page 3331)
  Indicates whether the window is removed from the screen when its application becomes inactive.
- `setHidesOnDeactivate:` (page 3386)
  Specifies whether the window is removed from the screen when the application is inactive.
- `collectionBehavior` (page 3309)
  Identifies the window's behavior in window collections.
- `setCollectionBehavior:` (page 3372)
  Specifies the window's behavior in window collections.
- `isOpaque` (page 3340)
  Indicates whether the window is opaque.
- `setOpaque:` (page 3392)
  Specifies whether the window is opaque.
- `hasShadow` (page 3331)
  Indicates whether the window has a shadow.
- `setHasShadow:` (page 3385)
  Specifies whether the window has a shadow.
- `invalidateShadow` (page 3336)
  Invalidates the window shadow so that it is recomputed based on the current window shape.
- `autorecalculatesContentBorderThicknessForEdge:` (page 3300)
  Indicates whether the window calculates the thickness of a given border automatically.
- `setAutorecalculatesContentBorderThickness:forEdge:` (page 3369)
  Specifies whether the window calculates the thickness of a given border automatically.
- `contentBorderThicknessForEdge:` (page 3310)
  Indicates the thickness of a given border of the window.
- `setContentBorderThickness:forEdge:` (page 3373)
  Specifies the thickness of a given border of the window.
- `delegate` (page 3316)
  Returns the window's delegate.
- `setDelegate:` (page 3377)
  Sets the window's delegate to a given object or removes an existing delegate.
- `preventsApplicationTerminationWhenModal` (page 3357)
  Indicates whether the window prevents application termination when modal.
- `setPreventsApplicationTerminationWhenModal:` (page 3394)
  Specifies whether the window prevents application termination when modal.

- `canBeVisibleOnAllSpaces` (page 3305) Deprecated in Mac OS X v10.5
  Indicates whether the window can be visible on all spaces or on only one space at a time. (Deprecated. Use `setCollectionBehavior:` (page 3372) instead.)
- `setCanBeVisibleOnAllSpaces:` (page 3371) Deprecated in Mac OS X v10.5
  Specifies whether the window can be visible on all spaces or on only one space at a time. (Deprecated. Use `collectionBehavior` (page 3309) instead.)

Accessing Window Information

+ `defaultDepthLimit` (page 3292)
  Returns the default depth limit for instances of `NSWindow`.
- `windowNumber` (page 3408)
  Provides the window number of the window's window device.
+ `windowNumbersWithOptions:` (page 3296)
  Returns the window numbers for all visible windows satisfying the specified options.
- `gState` (page 3330)
  Returns the window's graphics state object.
- `canStoreColor` (page 3306)
  Indicates whether the window has a depth limit that allows it to store color values.
- `deviceDescription` (page 3318)
  Returns a dictionary containing information about the window's resolution.
- `canBecomeVisibleWithoutLogin` (page 3305)
  Indicates whether the window can be displayed at the login window. Default: `NO`.
- `setCanBecomeVisibleWithoutLogin:` (page 3371)
  Specifies whether the window can be displayed at the login window.
- `sharingType` (page 3401)
  Indicates the level of access other processes have to the window's content.
- `setSharingType:` (page 3396)
  Specifies the level of access other processes have to the window's content.
- `backingType` (page 3302)
  Returns the window's backing store type.
- `setBackingType:` (page 3370)
  Sets the window's backing store type to a given type.
- `backingLocation` (page 3302)
  Indicates the window's backing store location.
- `preferredBackingLocation` (page 3356)
  Indicates the preferred location for the window's backing store.
- `setPreferredBackingLocation:` (page 3393)
  Specifies the preferred location for the window's backing store.
- `isOneShot` (page 3340)
  Indicates whether the window device the window manages is freed when it's removed from the screen list.

- `setOneShot:` (page 3391)
  Sets whether the window device that the window manages should be freed when it's removed from the screen list.
- `depthLimit` (page 3317)
  Returns the depth limit of the window.
- `setDepthLimit:` (page 3378)
  Sets the depth limit of the window to a given limit.
- `hasDynamicDepthLimit` (page 3330)
  Indicates whether the window's depth limit can change to match the depth of the screen it's on.
- `setDynamicDepthLimit:` (page 3380)
  Sets whether the window changes its depth to match the depth of the screen it's on, or the depth of the deepest screen when it spans multiple screens.

Getting Layout Information

+ `contentRectForFrameRect:styleMask:` (page 3292)
  Returns the content rectangle used by a window with a given frame rectangle and window style.
+ `frameRectForContentRect:styleMask:` (page 3293)
  Returns the frame rectangle used by a window with a given content rectangle and window style.
+ `minFrameWidthWithTitle:styleMask:` (page 3294)
  Returns the minimum width a window's frame rectangle must have for it to display a title, with a given window style.
- `contentRectForFrameRect:` (page 3312)
  Returns the window's content rectangle with a given frame rectangle.
- `frameRectForContentRect:` (page 3329)
  Returns the window's frame rectangle with a given content rectangle.

Managing Windows

- `drawers` (page 3324)
  Returns the collection of drawers associated with the window.
- `windowController` (page 3407)
  Returns the window's window controller.
- `setWindowController:` (page 3400)
  Sets the window's window controller.

Managing Sheets

- `attachedSheet` (page 3300)
  Returns the sheet attached to the window.
- `isSheet` (page 3341)
  Indicates whether the window has ever run as a modal sheet.

Sizing Windows

- `frame` (page 3328)
  Returns the window's frame rectangle.
- `setFrameOrigin:` (page 3383)
  Positions the bottom-left corner of the window's frame rectangle at a given point in screen coordinates.
- `setFrameTopLeftPoint:` (page 3384)
  Positions the top-left corner of the window's frame rectangle at a given point in screen coordinates.
- `constrainFrameRect:toScreen:` (page 3309)
  Modifies and returns a frame rectangle so that its top edge lies on a specific screen.
- `cascadeTopLeftFromPoint:` (page 3306)
  Positions the window's top left to a given point.
- `setFrame:display:` (page 3381)
  Sets the origin and size of the window's frame rectangle according to a given frame rectangle, thereby setting its position and size onscreen.
- `setFrame:display:animate:` (page 3381)
  Sets the origin and size of the window's frame rectangle, with optional animation, according to a given frame rectangle, thereby setting its position and size onscreen.
- `animationResizeTime:` (page 3299)
  Specifies the duration of a smooth frame-size change.
- `aspectRatio` (page 3300)
  Returns the window's aspect ratio, which constrains the size of its frame rectangle to integral multiples of this ratio when the user resizes it.
- `setAspectRatio:` (page 3368)
  Sets the window's aspect ratio, which constrains the size of its frame rectangle to integral multiples of this ratio when the user resizes it.
- `minSize` (page 3348)
  Returns the minimum size to which the window's frame (including its title bar) can be sized.
- `setMinSize:` (page 3390)
  Sets the minimum size to which the window's frame (including its title bar) can be sized to *aSize*.
- `maxSize` (page 3346)
  Returns the maximum size to which the window's frame (including its title bar) can be sized.
- `setMaxSize:` (page 3388)
  Sets the maximum size to which the window's frame (including its title bar) can be sized.
- `isZoomed` (page 3342)
  Returns a Boolean value that indicates whether the window is in a zoomed state.
- `performZoom:` (page 3355)
  This action method simulates the user clicking the zoom box by momentarily highlighting the button and then zooming the window.
- `zoom:` (page 3409)
  This action method toggles the size and location of the window between its standard state (provided by the application as the "best" size to display the window's data) and its user state (a new size and location the user may have set by moving or resizing the window).
- `resizeFlags` (page 3361)
  Returns the flags field of the event record for the mouse-down event that initiated the resizing session.

- `showsResizeIndicator` (page 3401)
  Returns a Boolean value that indicates whether the window's resize indicator is visible.
- `setShowsResizeIndicator:` (page 3397)
  Specifies whether the window's resize indicator is visible
- `resizeIncrements` (page 3361)
  Returns the window's resizing increments.
- `setResizeIncrements:` (page 3396)
  Restricts the user's ability to resize the window so the width and height change by multiples of width and height increments.
- `preservesContentDuringLiveResize` (page 3356)
  Returns whether the window tries to optimize user-initiated resize operations by preserving the content of views that have not changed.
- `setPreservesContentDuringLiveResize:` (page 3393)
  Specifies whether the window tries to optimize live resize operations by preserving the content of views that have not changed.
- `inLiveResize` (page 3335)
  Indicates whether the window is being resized by the user.

Sizing Content

- `contentAspectRatio` (page 3310)
  Returns the window's content aspect ratio.
- `setContentAspectRatio:` (page 3373)
  Sets the aspect ratio (height in relation to width) of the window's content view, constraining the dimensions of its content rectangle to integral multiples of that ratio when the user resizes it.
- `contentMinSize` (page 3311)
  Returns the minimum size of the window's content view.
- `setContentMinSize:` (page 3374)
  Sets the minimum size of the window's content view in the window's base coordinate system.
- `setContentSize:` (page 3375)
  Sets the size of the window's content view to a given size, which is expressed in the window's base coordinate system.
- `contentMaxSize` (page 3311)
  Returns the maximum size of the window's content view.
- `setContentMaxSize:` (page 3374)
  Sets the maximum size of the window's content view in the window's base coordinate system.
- `contentResizeIncrements` (page 3312)
  Returns the window's content-view resizing increments.
- `setContentResizeIncrements:` (page 3375)
  Restricts the user's ability to resize the window so the width and height of its content view change by multiples of width and height increments.

Managing Window Layers

- `isVisible` (page 3342)
  Indicates whether the window is visible onscreen (even when It's obscured by other windows).
- `orderOut:` (page 3352)
  Removes the window from the screen list, which hides the window.
- `orderBack:` (page 3350)
  Moves the window to the back of its level in the screen list, without changing either the key window or the main window.
- `orderFront:` (page 3351)
  Moves the window to the front of its level in the screen list, without changing either the key window or the main window.
- `orderFrontRegardless` (page 3351)
  Moves the window to the front of its level, even if its application isn't active, without changing either the key window or the main window.
- `orderWindow:relativeTo:` (page 3352)
  Repositions the window's window device in the window server's screen list.
- `level` (page 3343)
  Returns the window level of the window.
- `setLevel:` (page 3387)
  Sets the window's window level to a given level.

Managing Window Frames in User Defaults

+ `removeFrameUsingName:` (page 3294)
  Removes the frame data stored under a given name from the application's user defaults.
- `setFrameUsingName:` (page 3384)
  Sets the window's frame rectangle by reading the rectangle data stored under a given name from the defaults system.
- `setFrameUsingName:force:` (page 3385)
  Sets the window's frame rectangle by reading the rectangle data stored under a given name from the defaults system. Can operate on nonresizable windows.
- `saveFrameUsingName:` (page 3363)
  Saves the window's frame rectangle in the user defaults system under a given name.
- `frameAutosaveName` (page 3329)
  Returns the name used to automatically save the window's frame rectangle data in the defaults system, as set through `setFrameAutosaveName:` (page 3382).
- `setFrameAutosaveName:` (page 3382)
  Sets the name used to automatically save the window's frame rectangle in the defaults system to a given name.
- `stringWithSavedFrame` (page 3402)
  Returns a string representation of the window's frame rectangle.
- `setFrameFromString:` (page 3383)
  Sets the window's frame rectangle from a given string representation.

Managing Key Status

- `isKeyWindow` (page 3338)
  Indicates whether the window is the key window for the application.
- `canBecomeKeyWindow` (page 3304)
  Indicates whether the window can become the key window.
- `makeKeyWindow` (page 3345)
  Makes the window the key window.
- `makeKeyAndOrderFront:` (page 3345)
  Moves the window to the front of the screen list, within its level, and makes it the key window; that is, it shows the window.
- `becomeKeyWindow` (page 3302)
  Invoked automatically to inform the window that it has become the key window; never invoke this method directly.
- `resignKeyWindow` (page 3360)
  Invoked automatically when the window resigns key window status; never invoke this method directly.

Managing Main Status

- `isMainWindow` (page 3338)
  Indicates whether the window is the application's main window.
- `canBecomeMainWindow` (page 3304)
  Indicates whether the window can become the application's main window.
- `makeMainWindow` (page 3346)
  Makes the window the main window.
- `becomeMainWindow` (page 3303)
  Invoked automatically to inform the window that it has become the main window; never invoke this method directly.
- `resignMainWindow` (page 3361)
  Invoked automatically when the window resigns main window status; never invoke this method directly.

Managing Toolbars

- `toolbar` (page 3404)
  Returns the window's toolbar.
- `setToolbar:` (page 3399)
  Sets the window's toolbar.
- `toggleToolbarShown:` (page 3403)
  The action method for the "Hide Toolbar" menu item (which alternates with "Show Toolbar").
- `runToolbarCustomizationPalette:` (page 3362)
  The action method for the "Customize Toolbar…" menu item.

Managing Attached Windows

- `childWindows` (page 3308)
  Returns an array of the window's attached child windows.
- `addChildWindow:ordered:` (page 3297)
  Adds a given window as a child window of the window.
- `removeChildWindow:` (page 3358)
  Detaches a given child window from the window.
- `parentWindow` (page 3353)
  Returns the parent window to which the window is attached as a child.
- `setParentWindow:` (page 3392)
  Adds the window as a child of a given window. For use by subclasses when setting the parent window in the window.

Managing Window Buffers

- `isFlushWindowDisabled` (page 3337)
  Indicates whether the window's flushing ability is disabled.
- `enableFlushWindow` (page 3324)
  Reenables the `flushWindow` (page 3327) method for the window after it was disabled through a previous `disableFlushWindow` (page 3319) message.
- `disableFlushWindow` (page 3319)
  Disables the `flushWindow` (page 3327) method for the window.
- `flushWindow` (page 3327)
  Flushes the window's offscreen buffer to the screen if the window is buffered and flushing is enabled.
- `flushWindowIfNeeded` (page 3328)
  Flushes the window's offscreen buffer to the screen if flushing is enabled and if the last `flushWindow` (page 3327) message had no effect because flushing was disabled.

Managing Default Buttons

- `defaultButtonCell` (page 3316)
  Returns the button cell that performs as if clicked when the window receives a Return (or Enter) key event.
- `setDefaultButtonCell:` (page 3377)
  Makes the key equivalent of button cell the Return (or Enter) key, so when the user presses Return that button performs as if clicked.
- `enableKeyEquivalentForDefaultButtonCell` (page 3324)
  Reenables the default button cell's key equivalent, so it performs a click when the user presses Return (or Enter).
- `disableKeyEquivalentForDefaultButtonCell` (page 3319)
  Disables the default button cell's key equivalent, so it doesn't perform a click when the user presses Return (or Enter).

Managing Field Editors

- `fieldEditor:forObject:` (page 3325)
  Returns the window's field editor, creating it if requested.
- `endEditingFor:` (page 3325)
  Forces the field editor to give up its first responder status and prepares it for its next assignment.

Managing the Window Menu

- `isExcludedFromWindowsMenu` (page 3337)
  Indicates whether the window is excluded from the application's Windows menu.
- `setExcludedFromWindowsMenu:` (page 3380)
  Specifies whether the window's title is omitted from the application's Windows menu.

Managing Cursor Rectangles

- `areCursorRectsEnabled` (page 3299)
  Indicates whether the window's cursor rectangles are enabled.
- `enableCursorRects` (page 3324)
  Reenables cursor rectangle management within the window after a `disableCursorRects` (page 3318) message.
- `disableCursorRects` (page 3318)
  Disables all cursor rectangle management within the window.
- `discardCursorRects` (page 3320)
  Invalidates all cursor rectangles in the window.
- `invalidateCursorRectsForView:` (page 3335)
  Marks as invalid the cursor rectangles of a given `NSView` object in the window's view hierarchy, so they'll be set up again when the window becomes key (or immediately if the window is key).
- `resetCursorRects` (page 3360)
  Clears the window's cursor rectangles and the cursor rectangles of the `NSView` objects in its view hierarchy.

Managing Title Bars

+ `standardWindowButton:forStyleMask:` (page 3295)
  Returns a new instance of a given standard window button, sized appropriately for a given window style.
- `standardWindowButton:` (page 3402)
  Returns the window button of a given window button kind in the window's view hierarchy.
- `showsToolbarButton` (page 3401)
  Indicates whether the toolbar control button is currently displayed.
- `setShowsToolbarButton:` (page 3397)
  Specifies whether the window shows the toolbar control button.

Managing Tooltips

- `allowsToolTipsWhenApplicationIsInactive` (page 3298)
  Indicates whether the window can display tooltips even when the application is in the background.
- `setAllowsToolTipsWhenApplicationIsInactive:` (page 3367)
  Specifies whether the window can display tooltips even when the application is in the background.

Handling Events

+ `menuChanged:` (page 3294)
  This method does nothing; it is here for backward compatibility.
- `currentEvent` (page 3314)
  Returns the event currently being processed by the application, by invoking `NSApplication`'s `currentEvent` (page 142) method.
- `nextEventMatchingMask:` (page 3349)
  Returns the next event matching a given mask.
- `nextEventMatchingMask:untilDate:inMode:dequeue:` (page 3349)
  Forwards the message to the global `NSApplication` object, `NSApp`.
- `discardEventsMatchingMask:beforeEvent:` (page 3321)
  Forwards the message to the `NSApplication` object, `NSApp`.
- `postEvent:atStart:` (page 3355)
  Forwards the message to the global `NSApplication` object, `NSApp`.
- `sendEvent:` (page 3366)
  This action method dispatches mouse and keyboard events sent to the window by the `NSApplication` object.
- `tryToPerform:with:` (page 3404)
  Dispatches action messages with a given argument.

Managing Responders

- `initialFirstResponder` (page 3332)
  Returns view that's made first responder the first time the window is placed onscreen.
- `firstResponder` (page 3327)
  Returns the window's first responder.
- `setInitialFirstResponder:` (page 3387)
  Sets a given view as the one that's made first responder (also called the key view) the first time the window is placed onscreen.
- `makeFirstResponder:` (page 3344)
  Attempts to make a given responder the first responder for the window.

Managing the Key View Loop

- `selectKeyViewPrecedingView:` (page 3364)
  Makes key the view that precedes the given view.
- `selectKeyViewFollowingView:` (page 3364)
  Makes key the view that follows the given view.
- `selectPreviousKeyView:` (page 3365)
  This action method searches for a candidate previous key view and, if it finds one, invokes `makeFirstResponder:` (page 3344) to establish it as the first responder.
- `selectNextKeyView:` (page 3364)
  This action method searches for a candidate next key view and, if it finds one, invokes `makeFirstResponder:` (page 3344) to establish it as the first responder.
- `keyViewSelectionDirection` (page 3343)
  Returns the direction the window is currently using to change the key view.
- `autorecalculatesKeyViewLoop` (page 3301)
  Indicates whether the window automatically recalculates the key view loop when views are added.
- `recalculateKeyViewLoop` (page 3357)
  Marks the key view loop as dirty and in need of recalculation.
- `setAutorecalculatesKeyViewLoop:` (page 3370)
  Specifies whether to recalculate the key view loop automatically when views are added or removed.

Handling Keyboard Events

- `keyDown:` (page 3343)
  Handles a given keyboard event that may need to be interpreted as changing the key view or triggering a keyboard equivalent.

Handling Mouse Events

- `acceptsMouseMovedEvents` (page 3296)
  Indicates whether the window accepts mouse-moved events.
- `ignoresMouseEvents` (page 3331)
  Indicates whether the window is transparent to mouse events.
- `setIgnoresMouseEvents:` (page 3387)
  Specifies whether the window is transparent to mouse clicks and other mouse events, allowing overlay windows.
- `mouseLocationOutsideOfEventStream` (page 3348)
  Returns the current location of the pointer reckoned in the window's base coordinate system.
- `setAcceptsMouseMovedEvents:` (page 3366)
  Specifies whether the window is to accept mouse-moved events.
- `+ windowNumberAtPoint:belowWindowWithWindowNumber:` (page 3295)
  Returns the number of the frontmost window that would be hit by a mouse-down at the specified screen location.

Bracketing Drawing Operations

- `cacheImageInRect:` (page 3303)
  Stores the window's raster image from a given rectangle expressed in the window's base coordinate system.
- `restoreCachedImage` (page 3362)
  Splices the window's cached image rectangles, if any, back into its raster image (and buffer if it has one), undoing the effect of any drawing performed within those areas since they were established using `cacheImageInRect:` (page 3303).
- `discardCachedImage` (page 3320)
  Discards all of the window's cached image rectangles.

Drawing Windows

- `display` (page 3321)
  Passes a display message down the window's view hierarchy, thus redrawing all views within the window, including the frame view that draws the border, title bar, and other peripheral elements.
- `displayIfNeeded` (page 3321)
  Passes a `displayIfNeeded` message down the window's view hierarchy, thus redrawing all views that need to be displayed, including the frame view that draws the border, title bar, and other peripheral elements.
- `viewsNeedDisplay` (page 3407)
  Indicates whether any of the window's views need to be displayed.
- `setViewsNeedDisplay:` (page 3400)
  Specifies whether the window's views need to be displayed..
- `isAutodisplay` (page 3336)
  Indicates whether the window automatically displays views that need to be displayed.
- `setAutodisplay:` (page 3368)
  Specifies whether the window is to automatically display the views that are marked as needing it.
- `useOptimizedDrawing:` (page 3405)
  Specifies whether the window is to optimize focusing and drawing when displaying its views.
- `graphicsContext` (page 3330)
  Provides the graphics context associated with the window for the current thread.
- `allowsConcurrentViewDrawing` (page 3298)
  Indicates whether the window allows multithreaded view drawing.
- `setAllowsConcurrentViewDrawing:` (page 3366)
  Specifies whether the window allows its views to be drawn concurrently.

Updating Windows

- `disableScreenUpdatesUntilFlush` (page 3319)
  Disables the window's screen updates until the window is flushed.
- `update` (page 3405)
  Updates the window.

Dragging Items

- `dragImage:at:offset:event:pasteboard:source:slideBack:` (page 3323)
  Begins a dragging session.
- `registerForDraggedTypes:` (page 3358)
  Registers a give set of pasteboard types as the pasteboard types the window will accept as the destination of an image-dragging session.
- `unregisterDraggedTypes` (page 3405)
  Unregisters the window as a possible destination for dragging operations.

Converting Coordinates

- `convertBaseToScreen:` (page 3313)
  Converts a given point from the window's base coordinate system to the screen coordinate system.
- `convertScreenToBase:` (page 3314)
  Converts a given point from the screen coordinate system to the window's base coordinate system.
- `userSpaceScaleFactor` (page 3406)
  Returns the scale factor applied to the window.

Accessing Edited Status

- `isDocumentEdited` (page 3336)
  Indicates whether the window's document has been edited.
- `setDocumentEdited:` (page 3379)
  Specifies whether the window's document has been edited.

Managing Titles

- `title` (page 3403)
  Returns either the string that appears in the title bar of the window, or the path to the represented file.
- `setTitle:` (page 3398)
  Sets the string that appears in the window's title bar (if it has one) to a given string and displays the title.
- `setTitleWithRepresentedFilename:` (page 3399)
  Sets a given path as the window's title, formatting it as a file-system path, and records this path as the window's associated filename using `setRepresentedFilename:` (page 3395).
- `representedFilename` (page 3359)
  Returns the pathname of the file the window represents.
- `setRepresentedFilename:` (page 3395)
  Sets the pathname of the file the window represents.
- `representedURL` (page 3359)
  Provides the URL of the file the window represents.

- `setRepresentedURL:` (page 3395)
  Specifies the URL of the file the window represents.

Accessing Screen Information

- `screen` (page 3363)
  Returns the screen the window is on.
- `deepestScreen` (page 3315)
  Returns the deepest screen the window is on (it may be split over several screens).
- `displaysWhenScreenProfileChanges` (page 3322)
  Indicates whether the window context should be updated when the screen profile changes or when the window moves to a different screen.
- `setDisplaysWhenScreenProfileChanges:` (page 3378)
  Specifies whether the window context should be updated when the screen profile changes.

Moving Windows

- `isMovableByWindowBackground` (page 3339)
  Indicates whether the window is movable by clicking and dragging anywhere in its background.
- `setMovableByWindowBackground:` (page 3391)
  Sets whether the window is movable by clicking and dragging anywhere in its background.
- `isMovable` (page 3339)
  Indicates whether the window can be moved by clicking in its title bar or background.
- `setMovable:` (page 3391)
  Specifies whether the window can be dragged by clicking in its title bar or background.
- `center` (page 3307)
  Sets the window's location to the center of the screen.

Closing Windows

- `performClose:` (page 3354)
  This action method simulates the user clicking the close button by momentarily highlighting the button and then closing the window.
- `close` (page 3308)
  Removes the window from the screen.
- `isReleasedWhenClosed` (page 3341)
  Indicates whether the window is released when it receives the `close` message.
- `setReleasedWhenClosed:` (page 3394)
  Specifies whether the window is released when it receives the `close` message.

Minimizing Windows

- `isMiniaturized` (page 3338)
  Indicates whether the window is minimized.
- `performMiniaturize:` (page 3354)
  Simulates the user clicking the minimize button by momentarily highlighting the button, then minimizing the window.
- `miniaturize:` (page 3347)
  Removes the window from the screen list and displays the minimized window in the Dock.
- `deminiaturize:` (page 3317)
  Deminimizes the window.
- `miniwindowImage` (page 3347)
  Returns the custom miniaturized window image of the window.
- `setMiniwindowImage:` (page 3389)
  Sets the window's custom minimized window image to a given image.
- `miniwindowTitle` (page 3347)
  Returns the title displayed in the window's minimized window.
- `setMiniwindowTitle:` (page 3390)
  Sets the title of the window's miniaturized counterpart to a given string and redisplays it.

Getting the Dock Tile

- `dockTile` (page 3322)
  Provides the application's Dock tile.

Printing Windows

- `print:` (page 3357)
  This action method runs the Print panel, and if the user chooses an option other than canceling, prints the window (its frame view and all subviews).
- `dataWithEPSInsideRect:` (page 3314)
  Returns EPS data that draws the region of the window within a given rectangle.
- `dataWithPDFInsideRect:` (page 3315)
  Returns PDF data that draws the region of the window within a given rectangle.

Providing Services

- `validRequestorForSendType:returnType:` (page 3406)
  Searches for an object that responds to a Services request.

Working with Carbon

`- initWithWindowRef:` (page 3334)
Returns a Cocoa window created from a Carbon window.

`- windowRef` (page 3408)
Returns the Carbon `WindowRef` associated with the window, creating one if necessary.

Class Methods

contentRectForFrameRect:styleMask:

Returns the content rectangle used by a window with a given frame rectangle and window style.

```
+ (NSRect)contentRectForFrameRect:(NSRect)windowFrame
    styleMask:(NSUInteger)windowStyle
```

Parameters

windowFrame
The frame rectangle for the window expressed in screen coordinates.

windowStyle
The window style for the window. See "Constants" (page 3411) for a list of style mask values.

Return Value
The content rectangle, expressed in screen coordinates, used by the window with *windowFrame* and *windowStyle*.

Discussion
When a `NSWindow` instance is available, you should use `contentRectForFrameRect:` (page 3312) instead of this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ frameRectForContentRect:styleMask:` (page 3293)

Related Sample Code
GLUT

Declared In
`NSWindow.h`

defaultDepthLimit

Returns the default depth limit for instances of `NSWindow`.

```
+ (NSWindowDepth)defaultDepthLimit
```

Return Value
The default depth limit for instances of `NSWindow`, determined by the depth of the deepest screen level available to the window server.

Discussion
The value returned can be examined with the Application Kit functions `NSPlanarFromDepth` (page 3990), `NSColorSpaceFromDepth` (page 3966), `NSBitsPerSampleFromDepth` (page 3966), and `NSBitsPerPixelFromDepth` (page 3966).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- setDepthLimit:` (page 3378)
- `- setDynamicDepthLimit:` (page 3380)
- `- canStoreColor` (page 3306)

Declared In
`NSWindow.h`

frameRectForContentRect:styleMask:

Returns the frame rectangle used by a window with a given content rectangle and window style.

```
+ (NSRect)frameRectForContentRect:(NSRect)windowContentRect
    styleMask:(NSUInteger)windowStyle
```

Parameters

windowContentRect
The content rectangle for a window expressed in screen coordinates.

windowStyle
The window style for the window. See "Window Style Masks" (page 3411) for a list of style mask values.

Return Value
The frame rectangle, expressed in screen coordinates, used by the window with *windowContentRect* and *windowStyle*.

Discussion
When a `NSWindow` instance is available, you should use `frameRectForContentRect:` (page 3329) instead of this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `+ contentRectForFrameRect:styleMask:` (page 3292)

Related Sample Code
CocoaDragAndDrop
FunHouse
GLUT
OpenCL NBody Simulation Example

Declared In
NSWindow.h

menuChanged:

This method does nothing; it is here for backward compatibility.

```
+ (void)menuChanged:(NSMenu *)menu
```

Parameters

menu
The menu object that changed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- menu (page 2163) (NSResponder)

Declared In
NSWindow.h

minFrameWidthWithTitle:styleMask:

Returns the minimum width a window's frame rectangle must have for it to display a title, with a given window style.

```
+ (CGFloat)minFrameWidthWithTitle:(NSString *)windowTitle
    styleMask:(NSUInteger)windowStyle
```

Parameters

windowTitle
The title for the window.

windowStyle
The window style for the window. See "Window Style Masks" (page 3411) for a list of style mask values.

Return Value
The minimum width of the window's frame, using windowStyle, in order to display windowTitle.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

removeFrameUsingName:

Removes the frame data stored under a given name from the application's user defaults.

```
+ (void)removeFrameUsingName:(NSString *)frameName
```

Parameters

frameName

The name of the frame to remove.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setFrameUsingName: (page 3384)
- setFrameAutosaveName: (page 3382)

Declared In

NSWindow.h

standardWindowButton:forStyleMask:

Returns a new instance of a given standard window button, sized appropriately for a given window style.

```
+ (NSButton *)standardWindowButton:(NSWindowButton)windowButtonKind
    forStyleMask:(NSUInteger)windowStyle
```

Parameters

windowButtonKind

The kind of standard window button to return.

windowStyle

The window style for which *windowButtonKind* is to be sized. See "Window Style Masks" (page 3411) for the list of allowable values.

Return Value

The new window button of the kind identified by *windowButtonKind*; nil when no such button kind exists.

Discussion

The caller is responsible for adding the button to the view hierarchy and for setting the target to be the window.

Availability

Available in Mac OS X v10.2 and later.

See Also

- standardWindowButton: (page 3402)

Declared In

NSWindow.h

windowNumberAtPoint:belowWindowWithWindowNumber:

Returns the number of the frontmost window that would be hit by a mouse-down at the specified screen location.

```
+ (NSInteger)windowNumberAtPoint:(NSPoint)point
    belowWindowWithWindowNumber:(NSInteger)windowNumber
```

Parameters

point

The location of the mouse-down in screen coordinates.

windowNumber

If non-0, the search will start below *windowNumber* window in z-order.

Return Value

The window number of the window under the point. The window number returned may correspond to a window in another application.

Discussion

Because this method uses the same rules as mouse-down hit-testing, windows with transparency at the given point, and windows that ignore mouse events, will not be returned.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSWindow.h

windowNumbersWithOptions:

Returns the window numbers for all visible windows satisfying the specified options.

```
+ (NSArray *)windowNumbersWithOptions:(NSWindowNumberListOptions)options
```

Parameters

options

The possible options are specified in "NSWindowNumberListOptions" (page 3420).

Return Value

An array of window numbers for all visible windows satisfying the specified options.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSWindow.h

Instance Methods

acceptsMouseMovedEvents

Indicates whether the window accepts mouse-moved events.

```
- (BOOL)acceptsMouseMovedEvents
```

Return Value

YES when the window accepts (and distributes) mouse-moved events; otherwise, NO.

Discussion
The `NSWindow` default is `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAcceptsMouseMovedEvents:` (page 3366)

Declared In
`NSWindow.h`

addChildWindow:ordered:

Adds a given window as a child window of the window.

```
- (void)addChildWindow:(NSWindow *)childWindow
    ordered:(NSWindowOrderingMode)orderingMode
```

Parameters

childWindow
The child window to order.

orderingMode
`NSWindowAbove`: *childWindow* is ordered immediately in front of the window.
`NSWindowBelow`: *childWindow* is ordered immediately behind the window.

Discussion
After the *childWindow* is added as a child of the window, it is maintained in relative position indicated by *orderingMode* for subsequent ordering operations involving either window. While this attachment is active, moving *childWindow* will not cause the window to move (as in sliding a drawer in or out), but moving the window will cause *childWindow* to move.

Note that you should not create cycles between parent and child windows. For example, you should not add window B as child of window A, then add window A as a child of window B.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `removeChildWindow:` (page 3358)
- `childWindows` (page 3308)
- `parentWindow` (page 3353)
- `setParentWindow:` (page 3392)

Related Sample Code
CIAnnotation
MyMediaPlayer
TrackBall

Declared In
`NSWindow.h`

allowsConcurrentViewDrawing

Indicates whether the window allows multithreaded view drawing.

```
- (BOOL)allowsConcurrentViewDrawing
```

Return Value
`YES` if the window allows multithreaded view drawing; otherwise, `NO`.

Discussion
The default value is `YES`.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setAllowsConcurrentViewDrawing:` (page 3366)

Declared In
`NSWindow.h`

allowsToolTipsWhenApplicationIsInactive

Indicates whether the window can display tooltips even when the application is in the background.

```
- (BOOL)allowsToolTipsWhenApplicationIsInactive
```

Return Value
`YES` if the window can display tooltips even when the application is in the background; otherwise, `NO`.

Discussion
The default is `NO`.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setAllowsToolTipsWhenApplicationIsInactive:` (page 3367)

Declared In
`NSWindow.h`

alphaValue

Returns the window's alpha value.

```
- (CGFloat)alphaValue
```

Return Value
The window's alpha value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlphaValue:` (page 3367)

Related Sample Code
From A View to A Movie
FunkyOverlayWindow

Declared In
`NSWindow.h`

animationResizeTime:

Specifies the duration of a smooth frame-size change.

```
- (NSTimeInterval)animationResizeTime:(NSRect)newWindowFrame
```

Parameters
newWindowFrame
The frame rectangle specified in `setFrame:display:animate:` (page 3381).

Return Value
The duration of the frame size change.

Discussion
Subclasses can override this method to control the total time for the frame change.

The `NSWindow` implementation uses the value from the `NSWindowResizeTime` user default as the time in seconds to resize by 150 pixels. If this value is unspecified, `NSWindowResizeTime` is 0.20 seconds (this default value may be different in different releases of Mac OS X).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

areCursorRectsEnabled

Indicates whether the window's cursor rectangles are enabled.

```
- (BOOL)areCursorRectsEnabled
```

Return Value
`YES` when cursor rectangles are enabled; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `enableCursorRects` (page 3324)
- `addCursorRect:cursor:` (page 3139) (`NSView`)

Declared In
NSWindow.h

aspectRatio

Returns the window's aspect ratio, which constrains the size of its frame rectangle to integral multiples of this ratio when the user resizes it.

```
- (NSSize)aspectRatio
```

Return Value
The window's aspect ratio.

Discussion
The size of the window's frame rectangle is constrained to integral multiples of this ratio when the user resizes it. You can set an NSWindow object's size to any ratio programmatically.

Availability
Available in Mac OS X v10.0 and later.

See Also
- resizeIncrements (page 3361)
- setAspectRatio: (page 3368)
- setFrame:display: (page 3381)

Declared In
NSWindow.h

attachedSheet

Returns the sheet attached to the window.

```
- (NSWindow *)attachedSheet
```

Return Value
The sheet attached to the window; nil when the window doesn't have a sheet attached.

Availability
Available in Mac OS X v10.1 and later.

Declared In
NSWindow.h

autorecalculatesContentBorderThicknessForEdge:

Indicates whether the window calculates the thickness of a given border automatically.

```
- (BOOL)autorecalculatesContentBorderThicknessForEdge:(NSRectEdge)edge
```

Parameters

edge

Border whose thickness autorecalculation status to set:

- `NSMaxYEdge`: Top border.
- `NSMinYEdge`: Bottom border.

Return Value
`YES` when the window auto-recalculates the given border's thickness; otherwise, `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setAutorecalculatesContentBorderThickness:forEdge:` (page 3369)

Declared In
`NSWindow.h`

autorecalculatesKeyViewLoop

Indicates whether the window automatically recalculates the key view loop when views are added.

```
- (BOOL)autorecalculatesKeyViewLoop
```

Return Value
`YES` if the window automatically recalculates the key view loop when views are added; otherwise, `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- recalculateKeyViewLoop` (page 3357)
`- setAutorecalculatesKeyViewLoop:` (page 3370)

Declared In
`NSWindow.h`

backgroundColor

Returns the color of the window's background.

```
- (NSColor *)backgroundColor
```

Return Value
The window's background color.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setBackgroundColor:` (page 3370)

Declared In
NSWindow.h

backingLocation

Indicates the window's backing store location.

```
- (NSWindowBackingLocation)backingLocation
```

Return Value
The location of the window's backing store. See "NSWindowBackingLocation" (page 3420) for possible values.

Availability
Available in Mac OS X v10.5 and later.

See Also
- preferredBackingLocation (page 3356)

Declared In
NSWindow.h

backingType

Returns the window's backing store type.

```
- (NSBackingStoreType)backingType
```

Return Value
The backing store type.

Discussion
The possible return values are described in "NSBackingStoreType—Buffered Window Drawing" (page 3417).

Availability
Available in Mac OS X v10.0 and later.

See Also
- setBackingType: (page 3370)

Related Sample Code
StickiesWithCoreData

Declared In
NSWindow.h

becomeKeyWindow

Invoked automatically to inform the window that it has become the key window; never invoke this method directly.

```
- (void)becomeKeyWindow
```

Discussion
This method reestablishes the window's first responder, sends the `becomeKeyWindow` message to that object if it responds, and posts an `NSWindowDidBecomeKeyNotification` (page 3422) to the default notification center.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `makeKeyWindow` (page 3345)
- `makeKeyAndOrderFront:` (page 3345)
- `becomeMainWindow` (page 3303)

Declared In
`NSWindow.h`

becomeMainWindow

Invoked automatically to inform the window that it has become the main window; never invoke this method directly.

```
- (void)becomeMainWindow
```

Discussion
This method posts an `NSWindowDidBecomeMainNotification` (page 3423) to the default notification center.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `makeMainWindow` (page 3346)
- `becomeKeyWindow` (page 3302)

Declared In
`NSWindow.h`

cacheImageInRect:

Stores the window's raster image from a given rectangle expressed in the window's base coordinate system.

```
- (void)cacheImageInRect:(NSRect)rectangle
```

Parameters
rectangle
The rectangle representing the image to cache.

Discussion
This method allows the window to perform temporary drawing, such as a band around the selection as the user drags the mouse, and to quickly restore the previous image by invoking `restoreCachedImage` (page 3362) and `flushWindowIfNeeded` (page 3328). The next time the window displays, it discards its cached image rectangles. You can also explicitly use `discardCachedImage` (page 3320) to free the memory occupied by cached image rectangles. *aRect* is made integral before caching the image to avoid antialiasing artifacts.

Only the last cached rectangle is remembered and can be restored.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `display` (page 3321)

Declared In
`NSWindow.h`

canBecomeKeyWindow

Indicates whether the window can become the key window.

```
- (BOOL)canBecomeKeyWindow
```

Return Value
`YES` if the window can become the key window, otherwise, `NO`.

Discussion
Attempts to make the window the key window are abandoned if this method returns `NO`. The `NSWindow` (page 3275) implementation returns `YES` if the window has a title bar or a resize bar, or `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isKeyWindow` (page 3338)
- `makeKeyWindow` (page 3345)

Related Sample Code
CIAnnotation
FancyAbout
FunkyOverlayWindow
GLUT
RoundTransparentWindow

Declared In
`NSWindow.h`

canBecomeMainWindow

Indicates whether the window can become the application's main window.

```
- (BOOL)canBecomeMainWindow
```

Return Value
`YES` when the window can become the main window; otherwise, `NO`.

Discussion
Attempts to make the window the main window are abandoned if this method returns `NO`. The `NSWindow` implementation returns `YES` if the window is visible, is not an `NSPanel` object, and has a title bar or a resize mechanism. Otherwise it returns `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isMainWindow` (page 3338)
- `makeMainWindow` (page 3346)

Related Sample Code
FancyAbout

Declared In
`NSWindow.h`

canBecomeVisibleWithoutLogin

Indicates whether the window can be displayed at the login window. Default: `NO`.

```
- (BOOL)canBecomeVisibleWithoutLogin
```

Return Value
`YES` when the window can be displayed at the login window; otherwise, `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setCanBecomeVisibleWithoutLogin:` (page 3371)

Declared In
`NSWindow.h`

canBeVisibleOnAllSpaces

Indicates whether the window can be visible on all spaces or on only one space at a time. (Deprecated in Mac OS X v10.5. Use `setCollectionBehavior:` (page 3372) instead.)

```
- (BOOL)canBeVisibleOnAllSpaces
```

Return Value
`YES` when the window can be visible on all spaces; `NO` when it can be visible on only one space at a time.

Discussion
The default is `NO`.

Availability
Available in Mac OS X v10.5 and later.
Deprecated in Mac OS X v10.5.

Declared In
NSWindow.h

canHide

Indicates whether the window can be hidden when its application becomes hidden (during execution of the NSApplication hide: (page 148) method).

```
- (BOOL)canHide
```

Return Value
YES if the window can be hidden when its application becomes hidden; otherwise, NO.

Discussion
The default is YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setCanHide: (page 3372)

Declared In
NSWindow.h

canStoreColor

Indicates whether the window has a depth limit that allows it to store color values.

```
- (BOOL)canStoreColor
```

Return Value
YES when the window's depth limit allows it to store color values; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- depthLimit (page 3317)
- shouldDrawColor (page 3232) (NSView)

Declared In
NSWindow.h

cascadeTopLeftFromPoint:

Positions the window's top left to a given point.

```
- (NSPoint)cascadeTopLeftFromPoint:(NSPoint)topLeft
```

Parameters

topLeft

The new top-left point, in screen coordinates, for the window. When `NSZeroPoint`, the window is not moved, except as needed to constrain to the visible screen

Return Value
The point shifted from top left of the window in screen coordinates.

Discussion
The returned point can be passed to a subsequent invocation of `cascadeTopLeftFromPoint:` to position the next window so the title bars of both windows are fully visible.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setFrameTopLeftPoint:` (page 3384)

Related Sample Code
EnhancedDataBurn

Declared In
`NSWindow.h`

center

Sets the window's location to the center of the screen.

```
- (void)center
```

Discussion
The window is placed exactly in the center horizontally and somewhat above center vertically. Such a placement carries a certain visual immediacy and importance. This method doesn't put the window onscreen, however; use `makeKeyAndOrderFront:` (page 3345) to do that.

You typically use this method to place a window—most likely an alert dialog—where the user can't miss it. This method is invoked automatically when a panel is placed on the screen by the `runModalForWindow:` (page 163) method of the `NSApplication` class.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaVideoFrameToGWorld
From A View to A Movie
NewsReader
Sketch+Accessibility
ThreadsImportMovie

Declared In
`NSWindow.h`

childWindows

Returns an array of the window's attached child windows.

```
- (NSArray *)childWindows
```

Return Value
An array containing the window's child windows.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `removeChildWindow:` (page 3358)
- `addChildWindow:ordered:` (page 3297)
- `parentWindow` (page 3353)
- `setParentWindow:` (page 3392)

Declared In
`NSWindow.h`

close

Removes the window from the screen.

```
- (void)close
```

Discussion
If the window is set to be released when closed, a `release` message is sent to the object after the current event is completed. For an `NSWindow` object, the default is to be released on closing, while for an `NSPanel` object, the default is not to be released. You can use the `setReleasedWhenClosed:` (page 3394) method to change the default behavior.

A window doesn't have to be visible to receive the close message. For example, when the application terminates, it sends the close message to all windows in its window list, even those that are not currently visible.

The close method posts an `NSWindowWillCloseNotification` (page 3426) notification to the default notification center.

The close method differs in two important ways from the `performClose:` (page 3354) method:

- It does not attempt to send a `windowShouldClose:` (page 3937) message to the window or its delegate.
- It does not simulate the user clicking the close button by momentarily highlighting the button.

Use `performClose:` (page 3354) if you need these features.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `orderOut:` (page 3352)

Related Sample Code
ExtractMovieAudioToAIFF
From A View to A Movie
QTAudioContextInsert
QTAudioExtractionPanel
ThreadsImportMovie

Declared In
NSWindow.h

collectionBehavior

Identifies the window's behavior in window collections.

```
- (NSWindowCollectionBehavior)collectionBehavior
```

Return Value
The collection behavior identifier.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setCollectionBehavior: (page 3372)

Declared In
NSWindow.h

colorSpace

Returns the window's color space.

```
- (NSColorSpace *)colorSpace
```

Return Value
The color space. Will return nil If the window does not have a backing store, and is off-screen.

Availability
Available in Mac OS X v10.6 and later.

See Also
- setColorSpace: (page 3373)

Declared In
NSWindow.h

constrainFrameRect:toScreen:

Modifies and returns a frame rectangle so that its top edge lies on a specific screen.

```
- (NSRect)constrainFrameRect:(NSRect)frameRect toScreen:(NSScreen *)screen
```

Parameters

frameRect

The proposed frame rectangle to adjust.

screen

The screen on which the top edge of the window's frame is to lie.

Return Value
The adjusted frame rectangle.

Discussion
If the window is resizable and the window's height is greater than the screen height, the rectangle's height is adjusted to fit within the screen as well. The rectangle's width and horizontal location are unaffected. You shouldn't need to invoke this method yourself; it's invoked automatically (and the modified frame is used to locate and set the size of the window) whenever a titled `NSWindow` object is placed onscreen and whenever its size is changed.

Subclasses can override this method to prevent their instances from being constrained or to constrain them differently.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
iSpend

Declared In
`NSWindow.h`

contentAspectRatio

Returns the window's content aspect ratio.

```
- (NSSize)contentAspectRatio
```

Return Value
The window's content aspect ratio.

Discussion
The default content aspect ratio is `(0, 0)`.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setContentAspectRatio:` (page 3373)

Declared In
`NSWindow.h`

contentBorderThicknessForEdge:

Indicates the thickness of a given border of the window.

```
- (CGFloat)contentBorderThicknessForEdge:(NSRectEdge)edge
```

Parameters

edge

The border whose thickness to get:

- `NSMaxYEdge`: Top border.
- `NSMinYEdge`: Bottom border.

Return Value
Thickness of the given border, in points.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setContentBorderThickness:forEdge:` (page 3373)

Declared In
`NSWindow.h`

contentMaxSize

Returns the maximum size of the window's content view.

```
- (NSSize)contentMaxSize
```

Return Value
The maximum size of the window's content view.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setContentMaxSize:` (page 3374)
`- contentMinSize` (page 3311)

Declared In
`NSWindow.h`

contentMinSize

Returns the minimum size of the window's content view.

```
- (NSSize)contentMinSize
```

Return Value
The minimum size of the window's content view.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setContentMinSize:` (page 3374)
- `contentMaxSize` (page 3311)

Declared In
`NSWindow.h`

contentRectForFrameRect:

Returns the window's content rectangle with a given frame rectangle.

```
- (NSRect)contentRectForFrameRect:(NSRect)windowFrame
```

Parameters
windowFrame
The frame rectangle for the window expressed in screen coordinates.

Return Value
The window's content rectangle, expressed in screen coordinates, with *windowFrame*.

Discussion
The window uses its current style mask in computing the content rectangle. See "Window Style Masks" (page 3411) for a list of style mask values. The main advantage of this instance-method counterpart to `contentRectForFrameRect:styleMask:` (page 3292) is that it allows you to take toolbars into account when converting between content and frame rectangles. (The toolbar is not included in the content rectangle.)

Availability
Available in Mac OS X v10.3 and later.

See Also
- `frameRectForContentRect:` (page 3329)
+ `contentRectForFrameRect:styleMask:` (page 3292)

Related Sample Code
SampleRaster

Declared In
`NSWindow.h`

contentResizeIncrements

Returns the window's content-view resizing increments.

```
- (NSSize)contentResizeIncrements
```

Return Value
The window's content-view resizing increments.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setContentResizeIncrements:` (page 3375)

Declared In
NSWindow.h

contentView

Returns the window's content view, the highest accessible NSView object in the window's view hierarchy.

```
- (id)contentView
```

Return Value
The content view.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setContentView: (page 3376)

Related Sample Code
FunHouse
FunkyOverlayWindow
GLUT
MyMediaPlayer
VBL

Declared In
NSWindow.h

convertBaseToScreen:

Converts a given point from the window's base coordinate system to the screen coordinate system.

```
- (NSPoint)convertBaseToScreen:(NSPoint)point
```

Parameters
point
The point expressed in the window's base coordinate system.

Return Value
point expressed in screen coordinates.

Availability
Available in Mac OS X v10.0 and later.

See Also
- convertScreenToBase: (page 3314)
- convertPoint:toView: (page 3155) (NSView)

Related Sample Code
GLUT
Sketch+Accessibility
ZipBrowser

Declared In
NSWindow.h

convertScreenToBase:

Converts a given point from the screen coordinate system to the window's base coordinate system.

```
- (NSPoint)convertScreenToBase:(NSPoint)aPoint
```

Parameters

point

The point expressed in screen coordinates.

Return Value
point expressed in the window's base coordinate system.

Availability
Available in Mac OS X v10.0 and later.

See Also
- convertBaseToScreen: (page 3313)
- convertRect:fromView: (page 3157) (NSView)

Related Sample Code
GLUT
Sketch+Accessibility

Declared In
NSWindow.h

currentEvent

Returns the event currently being processed by the application, by invoking NSApplication's currentEvent (page 142) method.

```
- (NSEvent *)currentEvent
```

Return Value
The event being processed by the application.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

dataWithEPSInsideRect:

Returns EPS data that draws the region of the window within a given rectangle.

```
- (NSData *)dataWithEPSInsideRect:(NSRect)rect
```

Parameters

rect

A rectangle (expressed in the window's coordinate system) that identifies the region to be expressed as EPS data.

Return Value

The region in the window (identified by *rect*) as EPS data.

Discussion

This data can be placed on a pasteboard, written to a file, or used to create an `NSImage` object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `dataWithEPSInsideRect:` (page 3162) (`NSView`)
- `writeEPSInsideRect:toPasteboard:` (page 3248) (`NSView`)

Declared In

`NSWindow.h`

dataWithPDFInsideRect:

Returns PDF data that draws the region of the window within a given rectangle.

```
- (NSData *)dataWithPDFInsideRect:(NSRect)rect
```

Parameters

rect

A rectangle (expressed in the window's coordinate system) that identifies the region to be expressed as PDF data.

Return Value

The region in the window (identified by *rect*) as PDF data.

Discussion

This data can be placed on a pasteboard, written to a file, or used to create an `NSImage` object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `dataWithPDFInsideRect:` (page 3162) (`NSView`)
- `writePDFInsideRect:toPasteboard:` (page 3249) (`NSView`)

Declared In

`NSWindow.h`

deepestScreen

Returns the deepest screen the window is on (it may be split over several screens).

```
- (NSScreen *)deepestScreen
```

Return Value
The deepest screen the window is on; `nil` when the window is offscreen.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `screen` (page 3363)
+ `deepestScreen` (page 2314) (`NSScreen`)

Declared In
`NSWindow.h`

defaultButtonCell

Returns the button cell that performs as if clicked when the window receives a Return (or Enter) key event.

```
- (NSButtonCell *)defaultButtonCell
```

Return Value
The button cell.

Discussion
This cell draws itself as the focal element for keyboard interface control, unless another button cell is focused on, in which case the default button cell temporarily draws itself as normal and disables its key equivalent.

The window receives a Return key event if no responder in its responder chain claims it, or if the user presses the Control key along with the Return key.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDefaultButtonCell:` (page 3377)
- `disableKeyEquivalentForDefaultButtonCell` (page 3319)
- `enableKeyEquivalentForDefaultButtonCell` (page 3324)

Declared In
`NSWindow.h`

delegate

Returns the window's delegate.

```
- (id < NSWindowDelegate >)delegate
```

Return Value
The window's delegate, or `nil` if it doesn't have a delegate.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDelegate:` (page 3377)

Related Sample Code
FancyAbout
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSWindow.h`

deminiaturize:

Deminimizes the window.

```
- (void)deminiaturize:(id)sender
```

Parameters
sender
The message's sender.

Discussion
Invoke this method to programmatically deminimize a minimized window in the Dock.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `miniaturize:` (page 3347)
- `styleMask` (page 3402)

Related Sample Code
GLUT

Declared In
`NSWindow.h`

depthLimit

Returns the depth limit of the window.

```
- (NSWindowDepth)depthLimit
```

Return Value
Depth limit of the window.

Discussion
The value returned can be examined with the Application Kit functions `NSPlanarFromDepth` (page 3990), `NSColorSpaceFromDepth` (page 3966), `NSBitsPerSampleFromDepth` (page 3966), and `NSBitsPerPixelFromDepth` (page 3966).

Availability
Available in Mac OS X v10.0 and later.

See Also
+ defaultDepthLimit (page 3292)
- setDepthLimit: (page 3378)
- setDynamicDepthLimit: (page 3380)

Declared In
NSWindow.h

deviceDescription

Returns a dictionary containing information about the window's resolution.

```
- (NSDictionary *)deviceDescription
```

Return Value
A dictionary containing resolution information about the window, such as color, depth, and so on.

Discussion
This information is useful for tuning images and colors to the window's display capabilities. The contents of the dictionary are described in "Display Device—Descriptions" (page 3413).

Availability
Available in Mac OS X v10.0 and later.

See Also
- deviceDescription (page 2317) (NSScreen)
- bestRepresentationForDevice: (page 1337) (NSImage)
- colorUsingColorSpaceName: (page 700) (NSColor)

Related Sample Code
CocoaDVDPlayer

Declared In
NSWindow.h

disableCursorRects

Disables all cursor rectangle management within the window.

```
- (void)disableCursorRects
```

Discussion
Use this method when you need to do some special cursor manipulation and you don't want the Application Kit interfering.

Availability
Available in Mac OS X v10.0 and later.

See Also
- enableCursorRects (page 3324)

Declared In
NSWindow.h

disableFlushWindow

Disables the flushWindow (page 3327) method for the window.

```
- (void)disableFlushWindow
```

Discussion
If the window is buffered, disabling flushWindow (page 3327) prevents drawing from being automatically flushed by the NSView display... methods from the window's backing store to the screen. This method permits several views to be drawn before the results are shown to the user.

Flushing should be disabled only temporarily, while the window's display is being updated. Each disableFlushWindow message must be paired with a subsequent enableFlushWindow (page 3324) message. Invocations of these methods can be nested; flushing isn't reenabled until the last (unnested) enableFlushWindow (page 3324) message is sent.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaDVDPlayer

Declared In
NSWindow.h

disableKeyEquivalentForDefaultButtonCell

Disables the default button cell's key equivalent, so it doesn't perform a click when the user presses Return (or Enter).

```
- (void)disableKeyEquivalentForDefaultButtonCell
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- defaultButtonCell (page 3316)
- enableKeyEquivalentForDefaultButtonCell (page 3324)

Declared In
NSWindow.h

disableScreenUpdatesUntilFlush

Disables the window's screen updates until the window is flushed.

```
- (void)disableScreenUpdatesUntilFlush
```

Discussion
This method can be invoked to synchronize hardware surface flushes with the window's flushes. The window immediately disables screen updates using the `NSDisableScreenUpdates` (page 3969) function and reenables screen updates when the window flushes. Sending this message multiple times during a window update cycle has no effect.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
CocoaSlides
GLSLShowpiece

Declared In
`NSWindow.h`

discardCachedImage

Discards all of the window's cached image rectangles.

```
- (void)discardCachedImage
```

Discussion
An `NSWindow` object automatically discards its cached image rectangles when it displays.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `cacheImageInRect:` (page 3303)
- `restoreCachedImage` (page 3362)
- `display` (page 3321)

Declared In
`NSWindow.h`

discardCursorRects

Invalidates all cursor rectangles in the window.

```
- (void)discardCursorRects
```

Discussion
This method is invoked by `resetCursorRects` (page 3360) to clear out existing cursor rectangles before resetting them. You shouldn't invoke it in the code you write, but you might want to override it to change its behavior.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

discardEventsMatchingMask:beforeEvent:

Forwards the message to the `NSApplication` object, `NSApp`.

```
- (void)discardEventsMatchingMask:(NSUInteger)eventMask beforeEvent:(NSEvent
    *)lastEvent
```

Parameters

eventMask

The mask of the events to discard.

lastEvent

The event up to which queued events are discarded from the queue.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `discardEventsMatchingMask:beforeEvent:` (page 144) (`NSApplication`)

Declared In
`NSWindow.h`

display

Passes a display message down the window's view hierarchy, thus redrawing all views within the window, including the frame view that draws the border, title bar, and other peripheral elements.

```
- (void)display
```

Discussion
You rarely need to invoke this method. `NSWindow` objects normally record which of their views need display and display them automatically on each pass through the event loop.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `display` (page 3163) (`NSView`)
- `displayIfNeeded` (page 3321)
- `isAutodisplay` (page 3336)

Related Sample Code
GLUT

Declared In
`NSWindow.h`

displayIfNeeded

Passes a `displayIfNeeded` message down the window's view hierarchy, thus redrawing all views that need to be displayed, including the frame view that draws the border, title bar, and other peripheral elements.

```
- (void)displayIfNeeded
```

Discussion
This method is useful when you want to modify some number of views and then display only the ones that were modified.

You rarely need to invoke this method. `NSWindow` objects normally record which of their views need display and display them automatically on each pass through the event loop.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- display` (page 3321)
- `- displayIfNeeded` (page 3164) (`NSView`)
- `- setNeedsDisplay:` (page 3225) (`NSView`)
- `- isAutodisplay` (page 3336)

Related Sample Code
AnimatedTableView

Declared In
`NSWindow.h`

displaysWhenScreenProfileChanges

Indicates whether the window context should be updated when the screen profile changes or when the window moves to a different screen.

```
- (BOOL)displaysWhenScreenProfileChanges
```

Return Value
`YES` when the window context should be updated when the screen profile changes or when the window moves to a different screen; otherwise, `NO`.

Discussion
The default value is `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `- setDisplaysWhenScreenProfileChanges:` (page 3378)

Related Sample Code
VideoViewer

Declared In
`NSWindow.h`

dockTile

Provides the application's Dock tile.

```
- (NSDockTile *)dockTile
```

Return Value
The application's Dock tile.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSWindow.h

dragImage:at:offset:event:pasteboard:source:slideBack:

Begins a dragging session.

```
- (void)dragImage:(NSImage *)image at:(NSPoint)imageLocation
    offset:(NSSize)pointerOffset event:(NSEvent *)event pasteboard:(NSPasteboard
    *)pasteboard source:(id)sourceObject slideBack:(BOOL)slideBack
```

Parameters

image
The object to be dragged.

imageLocation
Location of the image's bottom-left corner in the window's coordinate system. It determines the placement of the dragged image under the pointer.

initialOffset
The pointer's location relative to the mouse-down location. Not used in Mac OS X v10.4 and later.

event
The left-mouse down event that triggered the dragging operation.

pasteboard
The pasteboard that holds the data to be transfered to the destination.

sourceObject
The object serving as the controller of the dragging operation. It must conform to the NSDraggingSource informal protocol.

slideBack
Specifies whether the drag image should slide back to *imageLocation* if it's rejected by the drag destination. Pass YES to specify slideback behavior or NO to specify that it should not.

Discussion
This method should be invoked only from within a view's implementation of the mouseDown: (page 2164) or mouseDragged: (page 2164) methods (which overrides the version defined in NSResponder class). Essentially the same as the NSView method of the same name, except that *imageLocation* is given in the NSWindow object's base coordinate system.

Availability
Available in Mac OS X v10.0 and later.

See Also
- dragImage:at:offset:event:pasteboard:source:slideBack: (page 3168) (NSView)

Declared In
NSWindow.h

drawers

Returns the collection of drawers associated with the window.

```
- (NSArray *)drawers
```

Return Value
The collection of drawers associated with the window.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDrawer.h

enableCursorRects

Reenables cursor rectangle management within the window after a `disableCursorRects` (page 3318) message.

```
- (void)enableCursorRects
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

enableFlushWindow

Reenables the `flushWindow` (page 3327) method for the window after it was disabled through a previous `disableFlushWindow` (page 3319) message.

```
- (void)enableFlushWindow
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaDVDPlayer

Declared In
NSWindow.h

enableKeyEquivalentForDefaultButtonCell

Reenables the default button cell's key equivalent, so it performs a click when the user presses Return (or Enter).

```
- (void)enableKeyEquivalentForDefaultButtonCell
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `defaultButtonCell` (page 3316)
- `disableKeyEquivalentForDefaultButtonCell` (page 3319)

Declared In
`NSWindow.h`

endEditingFor:

Forces the field editor to give up its first responder status and prepares it for its next assignment.

```
- (void)endEditingFor:(id)object
```

Parameters

object

The object that is using the window's field editor.

Discussion

If the field editor is the first responder, it's made to resign that status even if its `resignFirstResponder` (page 2189) method returns `NO`. This registration forces the field editor to send a `textDidEndEditing:` message to its delegate. The field editor is then removed from the view hierarchy, its delegate is set to `nil`, and it's emptied of any text it may contain.

This method is typically invoked by the object using the field editor when it's finished. Other objects normally change the first responder by simply using `makeFirstResponder:` (page 3344), which allows a field editor or other object to retain its first responder status if, for example, the user has entered an invalid value. The `endEditingFor:` (page 3325) method should be used only as a last resort if the field editor refuses to resign first responder status. Even in this case, you should always allow the field editor a chance to validate its text and take whatever other action it needs first. You can do this by first trying to make the `NSWindow` object the first responder:

```
if ([myWindow makeFirstResponder:myWindow]) {
    /* All fields are now valid; it's safe to use fieldEditor:forObject:
        to claim the field editor. */
}
else {
    /* Force first responder to resign. */
    [myWindow endEditingFor:nil];
}
```

Availability

Available in Mac OS X v10.0 and later.

See Also
- `fieldEditor:forObject:` (page 3325)
- `windowWillReturnFieldEditor:toObject:` (page 3940) (`NSWindowDelegate`)

Declared In
`NSWindow.h`

fieldEditor:forObject:

Returns the window's field editor, creating it if requested.

```
- (NSText *)fieldEditor:(BOOL)createWhenNeeded forObject:(id)anObject
```

Parameters

createWhenNeeded

If YES, creates a field editor if one doesn't exist; if NO, does not create a field editor.

A freshly created NSWindow object doesn't have a field editor. After a field editor has been created for a window, the *createWhenNeeded* argument is ignored. By passing NO for *createWhenNeeded* and testing the return value, however, you can predicate an action on the existence of the field editor.

anObject

A text-displaying object for which the delegate (in windowWillReturnFieldEditor:toObject: (page 3940)) assigns a custom field editor. Pass nil to get the default field editor, which can be the NSWindow field editor or a custom field editor returned by the delegate.

Return Value

Returns the field editor for the designated object (*anObject*) or, if *anObject* is nil, the default field editor. Returns nil if *createFlag* is NO and if the field editor doesn't exist.

Discussion

The field editor is a single NSTextView object that is shared among all the controls in a window for light text-editing needs. It is automatically instantiated when needed, and it can be used however your application sees fit. Typically, the field editor is used by simple text-bearing objects—for example, an NSTextField object uses its window's field editor to display and manipulate text. The field editor can be shared by any number of objects, and so its state may be constantly changing. Therefore, it shouldn't be used to display text that demands sophisticated layout (for this you should create a dedicated NSTextView object).

The field editor may be in use by some view object, so be sure to properly dissociate it from that object before actually using it yourself (the appropriate way to do this is illustrated in the description of endEditingFor: (page 3325)). Once you retrieve the field editor, you can insert it in the view hierarchy, set a delegate to interpret text events, and have it perform whatever editing is needed. Then, when it sends a textDidEndEditing: message to the delegate, you can get its text to display or store and remove the field editor using endEditingFor: (page 3325).

The window's delegate can substitute a custom field editor in place of the window's field editor by implementing windowWillReturnFieldEditor:toObject: (page 3940). The custom field editor can become the default editor (common to all text-displaying objects) or specific to a particular text-displaying object (*anObject*). The window sends this message to its delegate with itself and *anObject* as the arguments; if the delegate returns a non-nil value, the window returns that object instead of its field editor in fieldEditor:forObject:. However, note the following:

- If the window's delegate is identical to *anObject*, windowWillReturnFieldEditor:toObject: (page 3940) isn't sent to the delegate.
- The object returned by the delegate method, though it may become first responder, does not become the window's default field editor. Other objects continue to use the window's default field editor.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSWindow.h

firstResponder

Returns the window's first responder.

```
- (NSResponder *)firstResponder
```

Return Value
The first responder.

Discussion
The first responder is usually the first object in a responder chain to receive an event or action message. In most cases, the first responder is a view object in that the user selects or activates with the mouse or keyboard.

You can use the `firstResponder` method in custom subclasses of responder classes (`NSWindow`, `NSApplication`, `NSView`, and subclasses) to determine if an instance of the subclass is currently the first responder. You can also use it to help locate a text field that currently has first-responder status. For more on this subject, see "Event Handling Basics."

In Mac OS X v10.6 and later, *`firstResponder`* is key-value observing compliant.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `makeFirstResponder:` (page 3344)
- `acceptsFirstResponder` (page 2143) (`NSResponder`)

Related Sample Code
AnimatedTableView
GLUT

Declared In
`NSWindow.h`

flushWindow

Flushes the window's offscreen buffer to the screen if the window is buffered and flushing is enabled.

```
- (void)flushWindow
```

Discussion
Does nothing for other display devices, such as a printer. This method is automatically invoked by the `NSWindow` `display` (page 3321) and `displayIfNeeded` (page 3321) methods and the corresponding `NSView` `display` (page 3163) and `displayIfNeeded` (page 3164) methods.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `flushWindowIfNeeded` (page 3328)
- `disableFlushWindow` (page 3319)
- `enableFlushWindow` (page 3324)

Related Sample Code
QTKitFrameStepper

Declared In
NSWindow.h

flushWindowIfNeeded

Flushes the window's offscreen buffer to the screen if flushing is enabled and if the last `flushWindow` (page 3327) message had no effect because flushing was disabled.

```
- (void)flushWindowIfNeeded
```

Discussion
To avoid unnecessary flushing, use this method rather than `flushWindow` (page 3327) to flush an `NSWindow` object after flushing has been reenabled.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `disableFlushWindow` (page 3319)
- `enableFlushWindow` (page 3324)

Declared In
NSWindow.h

frame

Returns the window's frame rectangle.

```
- (NSRect)frame
```

Return Value
The frame rectangle of the window in screen coordinates, including the title bar.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setFrame:display:` (page 3381)
- `setFrame:display:animate:` (page 3381)

Related Sample Code
CIAnnotation
FunHouse
FunkyOverlayWindow
GLUT
Quartz Composer WWDC 2005 TextEdit

Declared In
NSWindow.h

frameAutosaveName

Returns the name used to automatically save the window's frame rectangle data in the defaults system, as set through `setFrameAutosaveName:` (page 3382).

```
- (NSString *)frameAutosaveName
```

Return Value
The name used to save the window's frame rectangle automatically in the defaults system.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setFrameUsingName:` (page 3384)

Declared In
`NSWindow.h`

frameRectForContentRect:

Returns the window's frame rectangle with a given content rectangle.

```
- (NSRect)frameRectForContentRect:(NSRect)windowContent
```

Parameters
windowContent
The content rectangle for the window expressed in screen coordinates.

Return Value
The window's frame rectangle, expressed in screen coordinates, with *windowContent*.

Discussion
The window uses its current style mask in computing the frame rectangle. See "Window Style Masks" (page 3411) for a list of style mask values. The major advantage of this instance-method counterpart to `frameRectForContentRect:styleMask:` (page 3293) is that it allows you to take toolbars into account when converting between content and frame rectangles. (The toolbar is included in the frame rectangle but not the content rectangle.)

Availability
Available in Mac OS X v10.3 and later.

See Also
`- contentRectForFrameRect:` (page 3312)
`+ frameRectForContentRect:styleMask:` (page 3293)

Related Sample Code
BasicCocoaAnimations
FunHouse

Declared In
`NSWindow.h`

graphicsContext

Provides the graphics context associated with the window for the current thread.

```
- (NSGraphicsContext *)graphicsContext
```

Return Value
The graphics context associated with the window for the current thread.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
CIAnnotation
FunHouse

Declared In
NSWindow.h

gState

Returns the window's graphics state object.

```
- (NSInteger)gState
```

Return Value
The graphics state object associated with the window.

Discussion
This graphics state is used by default for all `NSView` objects in the window's view hierarchy, but individual views can be made to use their own with the `NSView` method `allocateGState` (page 3146).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

hasDynamicDepthLimit

Indicates whether the window's depth limit can change to match the depth of the screen it's on.

```
- (BOOL)hasDynamicDepthLimit
```

Return Value
`YES` when the window has a dynamic depth limit; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setDynamicDepthLimit:` (page 3380)

Declared In
NSWindow.h

hasShadow

Indicates whether the window has a shadow.

```
- (BOOL)hasShadow
```

Return Value
YES when the window has a shadow; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setHasShadow: (page 3385)
- invalidateShadow (page 3336)

Declared In
NSWindow.h

hidesOnDeactivate

Indicates whether the window is removed from the screen when its application becomes inactive.

```
- (BOOL)hidesOnDeactivate
```

Return Value
YES if the window is removed from the screen when its application is deactivated; NO if it remains onscreen.

Discussion
The default for NSWindow is NO; the default for NSPanel is YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setHidesOnDeactivate: (page 3386)

Declared In
NSWindow.h

ignoresMouseEvents

Indicates whether the window is transparent to mouse events.

```
- (BOOL)ignoresMouseEvents
```

Return Value
YES when the window is transparent to mouse events; otherwise, NO.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `setIgnoresMouseEvents:` (page 3387)

Declared In
`NSWindow.h`

initialFirstResponder

Returns view that's made first responder the first time the window is placed onscreen.

```
- (NSView *)initialFirstResponder
```

Return Value
The view that's made first responder the first time the window is placed onscreen.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setInitialFirstResponder:` (page 3387)
- `setNextKeyView:` (page 3226) (`NSView`)

Declared In
`NSWindow.h`

initWithContentRect:styleMask:backing:defer:

Initializes the window with the specified values.

```
- (id)initWithContentRect:(NSRect)contentRect styleMask:(NSUInteger)windowStyle
    backing:(NSBackingStoreType)bufferingType defer:(BOOL)deferCreation
```

Parameters

contentRect
Location and size of the window's content area in screen coordinates. Note that the window server limits window position coordinates to ±16,000 and sizes to 10,000.

windowStyle
The window's style. It can be `NSBorderlessWindowMask`, or it can contain any of the options described in "Window Style Masks" (page 3411), combined using the C bitwise OR operator. Borderless windows display none of the usual peripheral elements and are generally useful only for display or caching purposes; you should normally not need to create them. Also, note that a window's style mask should include `NSTitledWindowMask` if it includes any of the others.

bufferingType
Specifies how the drawing done in the window is buffered by the window device, and possible values are described in "NSBackingStoreType—Buffered Window Drawing" (page 3417).

deferCreation

Specifies whether the window server creates a window device for the window immediately. When `YES`, the window server defers creating the window device until the window is moved onscreen. All display messages sent to the window or its views are postponed until the window is created, just before it's moved onscreen.

Return Value
The initialized window.

Discussion
This method is the designated initializer for the `NSWindow` class.

Deferring the creation of the window improves launch time and minimizes the virtual memory load on the window server.

The new window creates a view to be its default content view. You can replace it with your own object by using the `setContentView:` (page 3376) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `orderFront:` (page 3351)
- `setTitle:` (page 3398)
- `setOneShot:` (page 3391)
- `initWithContentRect:styleMask:backing:defer:screen:` (page 3333)

Related Sample Code
AnimatedTableView
FunkyOverlayWindow
LiveVideoMixer2
OpenCL NBody Simulation Example
VBL

Declared In
`NSWindow.h`

initWithContentRect:styleMask:backing:defer:screen:

Initializes an allocated window with the specified values.

```
- (id)initWithContentRect:(NSRect)contentRect styleMask:(NSUInteger)windowStyle
    backing:(NSBackingStoreType)bufferingType defer:(BOOL)deferCreation
    screen:(NSScreen *)screen
```

Parameters

contentRect

Location and size of the window's content area in screen coordinates. Note that the window server limits window position coordinates to ±16,000 and sizes to 10,000.

windowStyle

The window's style. It can be either `NSBorderlessWindowMask`, or it can contain any of the options described in "Window Style Masks" (page 3411), combined using the C bitwise OR operator. Borderless windows display none of the usual peripheral elements and are generally useful only for display or caching purposes; you should normally not need to create them. Also, note that a window's style mask should include `NSTitledWindowMask` if it includes any of the others.

bufferingType

Specifies how the drawing done in the window is buffered by the window device; possible values are described in "NSBackingStoreType—Buffered Window Drawing" (page 3417).

deferCreation

Specifies whether the window server creates a window device for the window immediately. When `YES`, the window server defers creating the window device until the window is moved onscreen. All display messages sent to the window or its views are postponed until the window is created, just before it's moved onscreen.

screen

Specifies where the window's content rectangle is drawn if the window is to be drawn in a screen other than the main screen. The content rectangle is drawn relative to the bottom-left corner of `screen`. When `nil`, the content rectangle is drawn on the main screen.

Return Value
The initialized window.

Discussion
The main screen is the one that contains the current key window or, if there is no key window, the one that contains the main menu. If there's neither a key window nor a main menu (if there's no active application), the main screen is the one where the origin of the screen coordinate system is located.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- orderFront:` (page 3351)
- `- setTitle:` (page 3398)
- `- setOneShot:` (page 3391)

Related Sample Code
WhackedTV

Declared In
`NSWindow.h`

initWithWindowRef:

Returns a Cocoa window created from a Carbon window.

```
- (NSWindow *)initWithWindowRef:(void *)carbonWindowRef
```

Parameters

carbonWindowRef

The Carbon `WindowRef` object to use to create the Cocoa window.

Return Value
A Cocoa window created from *carbonWindowRef*.

Discussion
For more information on Carbon-Cocoa integration, see Using a Carbon User Interface in a Cocoa Application in *Carbon-Cocoa Integration Guide*.

Special Considerations

For historical reasons, contrary to normal memory management policy `initWithWindowRef:` does *not* retain *`windowRef`*. It is therefore recommended that you make sure you retain *`windowRef`* before calling this method. If *`windowRef`* is still valid when the Cocoa window is deallocated, the Cocoa window will release it.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `windowRef` (page 3408)

Related Sample Code
CarbonCocoaTempConverter

Declared In
`NSWindow.h`

inLiveResize

Indicates whether the window is being resized by the user.

```
- (BOOL)inLiveResize
```

Return Value
`YES` if the window is being live resized; otherwise, `NO`.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSWindow.h`

invalidateCursorRectsForView:

Marks as invalid the cursor rectangles of a given `NSView` object in the window's view hierarchy, so they'll be set up again when the window becomes key (or immediately if the window is key).

```
- (void)invalidateCursorRectsForView:(NSView *)view
```

Parameters

`view`
The view in the window's view hierarchy.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `resetCursorRects` (page 3360)

- resetCursorRects (page 3206) (NSView)

Related Sample Code
DragItemAround
GLUT

Declared In
NSWindow.h

invalidateShadow

Invalidates the window shadow so that it is recomputed based on the current window shape.

```
- (void)invalidateShadow
```

Availability
Available in Mac OS X v10.2 and later.

See Also
- hasShadow (page 3331)
- setHasShadow: (page 3385)

Declared In
NSWindow.h

isAutodisplay

Indicates whether the window automatically displays views that need to be displayed.

```
- (BOOL)isAutodisplay
```

Return Value
YES when the window automatically displays views that need to be displayed; otherwise, NO.

Discussion
Automatic display typically occurs on each pass through the event loop.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setAutodisplay: (page 3368)
- displayIfNeeded (page 3321)
- setNeedsDisplay: (page 3225) (NSView)

Declared In
NSWindow.h

isDocumentEdited

Indicates whether the window's document has been edited.

```
- (BOOL)isDocumentEdited
```

Return Value
`YES` when the window's document has been edited; otherwise, `NO`.

Discussion
Initially, by default, `NSWindow` objects are in the "not edited" state.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

isExcludedFromWindowsMenu

Indicates whether the window is excluded from the application's Windows menu.

```
- (BOOL)isExcludedFromWindowsMenu
```

Return Value
`YES` when the window is excluded from the Windows menu; otherwise, `NO`.

Discussion
The default initial setting is `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setExcludedFromWindowsMenu:` (page 3380)

Declared In
`NSWindow.h`

isFlushWindowDisabled

Indicates whether the window's flushing ability is disabled.

```
- (BOOL)isFlushWindowDisabled
```

Return Value
`YES` when the window's flushing ability has been disabled; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- disableFlushWindow` (page 3319)
`- enableFlushWindow` (page 3324)

Declared In
`NSWindow.h`

isKeyWindow

Indicates whether the window is the key window for the application.

```
- (BOOL)isKeyWindow
```

Return Value
`YES` if the window is the key window for the application; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isMainWindow` (page 3338)
- `makeKeyWindow` (page 3345)

Related Sample Code
AnimatedTableView

Declared In
`NSWindow.h`

isMainWindow

Indicates whether the window is the application's main window.

```
- (BOOL)isMainWindow
```

Return Value
`YES` when the window is the main window for the application; otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `isKeyWindow` (page 3338)
- `makeMainWindow` (page 3346)

Declared In
`NSWindow.h`

isMiniaturized

Indicates whether the window is minimized.

```
- (BOOL)isMiniaturized
```

Return Value
`YES` if the window is minimized; otherwise, `NO`.

Discussion
A minimized window is removed from the screen and replaced by a image, icon, or button that represents it, called the counterpart.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `miniaturize:` (page 3347)

Declared In
`NSWindow.h`

isMovable

Indicates whether the window can be moved by clicking in its title bar or background.

```
- (BOOL)isMovable
```

Return Value
`YES` if the window can be moved by the user; otherwise, `NO`.

Discussion
`setMovableByWindowBackground:` (page 3391), called with the argument `YES`, is ignored by a window that returns `NO` from `isMovable`. If a window returns `NO`, that means it can only be dragged between spaces in F8 mode, and its relative screen position is always preserved. Note that a resizable window may still be resized, and the window frame may be changed programmatically. A non-movable window will not be moved or resized by the system in response to a display reconfiguration. Applications may choose to enable application-controlled window dragging after disabling user-initiating dragging by handling the `mouseDown:` (page 2164)/`mouseDragged:` (page 2164)/`mouseUp:` (page 2166) sequence in `sendEvent:` (page 3366) in an `NSWindow` subclass.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `setMovable:` (page 3391)

Declared In
`NSWindow.h`

isMovableByWindowBackground

Indicates whether the window is movable by clicking and dragging anywhere in its background.

```
- (BOOL)isMovableByWindowBackground
```

Return Value
`YES` when the window is movable by clicking and dragging anywhere in its background; otherwise, `NO`.

Discussion
A window with a style mask of `NSTexturedBackgroundWindowMask` is movable by background by default. Sheets and drawers cannot be movable by window background.

Availability
Available in Mac OS X v10.2 and later.

See Also
- setMovableByWindowBackground: (page 3391)

Declared In
NSWindow.h

isOnActiveSpace

Indicates whether the window is on the currently active space.

```
- (BOOL)isOnActiveSpace
```

Return Value
YES if the window is on the currently active space; otherwise, NO.

Discussion
For visible windows, this method indicates whether the window is currently visible on the active space. For offscreen windows, it indicates whether ordering the window onscreen would cause it to be on the active space.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSWindow.h

isOneShot

Indicates whether the window device the window manages is freed when it's removed from the screen list.

```
- (BOOL)isOneShot
```

Return Value
YES when the window's window device is freed when it's removed from the screen list; otherwise, NO.

Discussion
The default is NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setOneShot: (page 3391)

Declared In
NSWindow.h

isOpaque

Indicates whether the window is opaque.

```
- (BOOL)isOpaque
```

Return Value
YES when the window is opaque; otherwise, NO.

Discussion
The default is YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setOpaque: (page 3392)

Declared In
NSWindow.h

isReleasedWhenClosed

Indicates whether the window is released when it receives the close message.

```
- (BOOL)isReleasedWhenClosed
```

Return Value
YES if the window is automatically released after being closed; NO if it's simply removed from the screen.

Discussion
The default for NSWindow is YES; the default for NSPanel is NO. Release when closed, however, is ignored for windows owned by window controllers.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setReleasedWhenClosed: (page 3394)

Declared In
NSWindow.h

isSheet

Indicates whether the window has ever run as a modal sheet.

```
- (BOOL)isSheet
```

Return Value
YES if the window has ever run as a modal sheet; otherwise, NO.

Discussion
Sheets are created using the NSPanel subclass.

Availability
Available in Mac OS X v10.1 and later.

Declared In
NSWindow.h

isVisible

Indicates whether the window is visible onscreen (even when It's obscured by other windows).

```
- (BOOL)isVisible
```

Return Value
`YES` when the window is onscreen (even if it's obscured by other windows); otherwise, `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- visibleRect` (page 3245) (`NSView`)

Related Sample Code
AnimatedTableView
GLUT
QTAudioExtractionPanel
Sketch+Accessibility

Declared In
`NSWindow.h`

isZoomed

Returns a Boolean value that indicates whether the window is in a zoomed state.

```
- (BOOL)isZoomed
```

Return Value
`YES` if the window is in a zoomed state; otherwise, `NO`.

Discussion
The zoomed state of the window is determined using the following steps:

1. If the delegate or the window class implements `windowWillUseStandardFrame:defaultFrame:` (page 3942), it is invoked to obtain the zoomed frame of the window. The result of `isZoomed` is then determined by whether or not the current window frame is equal to the zoomed frame.

2. If the neither the delegate nor the window class implements `windowWillUseStandardFrame:defaultFrame:`, a default frame that nearly fits the screen is chosen. If the delegate or window class implements , it is invoked to validate the proposed zoomed frame. Once the zoomed frame is validated, the result of `isZoomed` is determined by whether or not the current window frame is equal to the zoomed frame.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- zoom:` (page 3409)

Declared In
NSWindow.h

keyDown:

Handles a given keyboard event that may need to be interpreted as changing the key view or triggering a keyboard equivalent.

```
- (void)keyDown:(NSEvent *)event
```

Parameters

event
The keyboard event to process.

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectNextKeyView: (page 3364)
- nextKeyView (page 3192) (NSView)
- performMnemonic: (page 3195) (NSView)

Declared In
NSWindow.h

keyViewSelectionDirection

Returns the direction the window is currently using to change the key view.

```
- (NSSelectionDirection)keyViewSelectionDirection
```

Return Value
The direction the window is using to change the key view.

Discussion
This direction can be one of the values described in "NSSelectionDirection—Direction of Key View Change" (page 3415).

Availability
Available in Mac OS X v10.0 and later.

See Also
- selectNextKeyView: (page 3364)
- selectPreviousKeyView: (page 3365)

Declared In
NSWindow.h

level

Returns the window level of the window.

```
- (NSInteger)level
```

Return Value
The window level.

Discussion
See "Window Levels" (page 3412) for a list of possible values.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setLevel:` (page 3387)

Declared In
`NSWindow.h`

makeFirstResponder:

Attempts to make a given responder the first responder for the window.

```
- (BOOL)makeFirstResponder:(NSResponder *)responder
```

Parameters

responder
The responder to set as the window's first responder. `nil` makes the window its first responder.

Return Value
`YES` when the operation is successful; otherwise, `NO`.

Discussion
If *responder* isn't already the first responder, this method first sends a `resignFirstResponder` (page 2189) message to the object that is the first responder. If that object refuses to resign, it remains the first responder, and this method immediately returns `NO`. If the current first responder resigns, this method sends a `becomeFirstResponder` (page 2144) message to *responder*. If *responder* does not accept first responder status, the `NSWindow` object becomes first responder; in this case, the method returns `YES` even if *responder* refuses first responder status.

If *responder* is `nil`, this method still sends `resignFirstResponder` (page 2189) to the current first responder. If the current first responder refuses to resign, it remains the first responder and this method immediately returns `NO`. If the current first responder returns `YES` from `resignFirstResponder`, the window is made its own first responder and this method returns `YES`.

The Application Kit framework uses this method to alter the first responder in response to mouse-down events; you can also use it to explicitly set the first responder from within your program. The *responder* object is typically an `NSView` object in the window's view hierarchy. If this method is called explicitly, first send `acceptsFirstResponder` (page 2143) to *responder*, and do not call `makeFirstResponder:` if `acceptsFirstResponder` returns `NO`.

Use `setInitialFirstResponder:` (page 3387) to the set the first responder to be used when the window is brought onscreen for the first time.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `becomeFirstResponder` (page 2144) (`NSResponder`)
- `resignFirstResponder` (page 2189) (`NSResponder`)

Related Sample Code
CocoaSpeechSynthesisExample
IdentitySample
ObjectPath
Sketch-112
WhackedTV

Declared In
`NSWindow.h`

makeKeyAndOrderFront:

Moves the window to the front of the screen list, within its level, and makes it the key window; that is, it shows the window.

```
- (void)makeKeyAndOrderFront:(id)sender
```

Parameters
sender
The message's sender.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `orderFront:` (page 3351)
- `orderBack:` (page 3350)
- `orderOut:` (page 3352)
- `orderWindow:relativeTo:` (page 3352)
- `setLevel:` (page 3387)

Related Sample Code
GLUT
GridCalendar
QTAudioExtractionPanel
QTKitPlayer
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSWindow.h`

makeKeyWindow

Makes the window the key window.

```
- (void)makeKeyWindow
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `makeMainWindow` (page 3346)
- `becomeKeyWindow` (page 3302)
- `isKeyWindow` (page 3338)

Declared In
`NSWindow.h`

makeMainWindow

Makes the window the main window.

```
- (void)makeMainWindow
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `makeKeyWindow` (page 3345)
- `becomeMainWindow` (page 3303)
- `isMainWindow` (page 3338)

Declared In
`NSWindow.h`

maxSize

Returns the maximum size to which the window's frame (including its title bar) can be sized.

```
- (NSSize)maxSize
```

Return Value
The maximum size to which the window's frame (including its title bar) can be sized either by the user or by the `setFrame...` methods other than `setFrame:display:` (page 3381) and `setFrame:display:animate:` (page 3381).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMaxSize:` (page 3388)
- `minSize` (page 3348)
- `aspectRatio` (page 3300)
- `resizeIncrements` (page 3361)

Declared In
`NSWindow.h`

miniaturize:

Removes the window from the screen list and displays the minimized window in the Dock.

```
- (void)miniaturize:(id)sender
```

Parameters

sender

The message's sender.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `deminiaturize:` (page 3317)

Related Sample Code
GLUT
StickiesWithCoreData

Declared In
`NSWindow.h`

miniwindowImage

Returns the custom miniaturized window image of the window.

```
- (NSImage *)miniwindowImage
```

Return Value
The custom miniaturized window image.

Discussion
The miniaturized window image is the image displayed in the Dock when the window is minimized. If you did not assign a custom image to the window, this method returns `nil`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMiniwindowImage:` (page 3389)
- `miniwindowTitle` (page 3347)

Declared In
`NSWindow.h`

miniwindowTitle

Returns the title displayed in the window's minimized window.

```
- (NSString *)miniwindowTitle
```

Return Value
The title displayed in the window's minimized window.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMiniwindowTitle:` (page 3390)
- `miniwindowImage` (page 3347)

Declared In
`NSWindow.h`

minSize

Returns the minimum size to which the window's frame (including its title bar) can be sized.

```
- (NSSize)minSize
```

Return Value
The minimum size to which the window's frame (including its title bar) can be sized either by the user or by the `setFrame...` methods other than `setFrame:display:` (page 3381) and `setFrame:display:animate:` (page 3381).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMinSize:` (page 3390)
- `maxSize` (page 3346)
- `aspectRatio` (page 3300)
- `resizeIncrements` (page 3361)

Related Sample Code
CIVideoDemoGL
StickiesWithCoreData

Declared In
`NSWindow.h`

mouseLocationOutsideOfEventStream

Returns the current location of the pointer reckoned in the window's base coordinate system.

```
- (NSPoint)mouseLocationOutsideOfEventStream
```

Return Value
The current location of the pointer reckoned in the window's base coordinate system, regardless of the current event being handled or of any events pending.

Discussion
For the same information in screen coordinates, use `NSEvent`'s `mouseLocation` (page 1067).

Availability
Available in Mac OS X v10.0 and later.

See Also
- currentEvent (page 142) (NSApplication)

Related Sample Code
GLUT

Declared In
NSWindow.h

nextEventMatchingMask:

Returns the next event matching a given mask.

```
- (NSEvent *)nextEventMatchingMask:(NSUInteger)eventMask
```

Parameters

eventMask

The mask that the event to return must match. Events with nonmatching masks are removed from the queue. See discardEventsMatchingMask:beforeEvent: (page 144) in NSApplication for the list of mask values.

Return Value
The next event whose mask matches *eventMask*; nil when no matching event was found.

Discussion
This method sends the message nextEventMatchingMask:eventMask untilDate:[NSDate distantFuture] inMode:NSEventTrackingRunLoopMode dequeue:YES to the application (NSApp).

Availability
Available in Mac OS X v10.0 and later.

See Also
- nextEventMatchingMask:untilDate:inMode:dequeue: (page 153) (NSApplication)

Related Sample Code
PhotoSearch
Sketch-112

Declared In
NSWindow.h

nextEventMatchingMask:untilDate:inMode:dequeue:

Forwards the message to the global NSApplication object, NSApp.

```
- (NSEvent *)nextEventMatchingMask:(NSUInteger)eventMask untilDate:(NSDate
    *)expirationDate inMode:(NSString *)runLoopMode dequeue:(BOOL)dequeue
```

Parameters

eventMask

The mask that the event to return must match.

expirationDate
The date until which to wait for events.

runLoopMode
The run loop mode to use while waiting for events

dequeue
YES to remove the returned event from the event queue; NO to leave the returned event in the queue.

Return Value
The next event whose mask matches *eventMask*; nil when no matching event was found.

Availability
Available in Mac OS X v10.0 and later.

See Also
- nextEventMatchingMask:untilDate:inMode:dequeue: (page 153) (NSApplication)

Related Sample Code
CIAnnotation
LiveVideoMixer
LiveVideoMixer2
LiveVideoMixer3
ThreadsExportMovie

Declared In
NSWindow.h

orderBack:

Moves the window to the back of its level in the screen list, without changing either the key window or the main window.

```
- (void)orderBack:(id)sender
```

Parameters

sender
Message originator.

Availability
Available in Mac OS X v10.0 and later.

See Also
- orderFront: (page 3351)
- orderOut: (page 3352)
- orderWindow:relativeTo: (page 3352)
- makeKeyAndOrderFront: (page 3345)
- level (page 3343)

Related Sample Code
GLUT

Declared In
NSWindow.h

orderFront:

Moves the window to the front of its level in the screen list, without changing either the key window or the main window.

```
- (void)orderFront:(id)sender
```

Parameters

sender

The message's sender.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `orderBack:` (page 3350)
- `orderOut:` (page 3352)
- `orderWindow:relativeTo:` (page 3352)
- `makeKeyAndOrderFront:` (page 3345)
- `level` (page 3343)

Related Sample Code

AnimatedTableView
EnhancedDataBurn
FunkyOverlayWindow
PreLoginAgents
QTQuartzPlayer

Declared In

`NSWindow.h`

orderFrontRegardless

Moves the window to the front of its level, even if its application isn't active, without changing either the key window or the main window.

```
- (void)orderFrontRegardless
```

Parameters

sender

The message's sender.

Discussion

Normally an `NSWindow` object can't be moved in front of the key window unless it and the key window are in the same application. You should rarely need to invoke this method; it's designed to be used when applications are cooperating in such a way that an active application (with the key window) is using another application to display data.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `orderFront:` (page 3351)

- level (page 3343)

Related Sample Code
PreLoginAgents

Declared In
NSWindow.h

orderOut:

Removes the window from the screen list, which hides the window.

```
- (void)orderOut:(id)sender
```

Parameters

sender

The message's sender.

Discussion
If the window is the key or main window, the NSWindow object immediately behind it is made key or main in its place. Calling the orderOut: (page 3352) method causes the window to be removed from the screen, but does not cause it to be released. See the close (page 3308) method for information on when a window is released.

Availability
Available in Mac OS X v10.0 and later.

See Also
- orderFront: (page 3351)
- orderBack: (page 3350)
- orderWindow:relativeTo: (page 3352)
- setReleasedWhenClosed: (page 3394)

Related Sample Code
EnhancedAudioBurn
EnhancedDataBurn
GridCalendar
ImageClient
Quartz Composer WWDC 2005 TextEdit

Declared In
NSWindow.h

orderWindow:relativeTo:

Repositions the window's window device in the window server's screen list.

```
- (void)orderWindow:(NSWindowOrderingMode)orderingMode
    relativeTo:(NSInteger)otherWindowNumber
```

Parameters

orderingMode

`NSWindowOut` (page 3419): The window is removed from the screen list and *otherWindowNumber* is ignored.

`NSWindowAbove` (page 3419): The window is ordered immediately in front of the window whose window number is *otherWindowNumber*

`NSWindowBelow` (page 3419): The window is placed immediately behind the window represented by *otherWindowNumber*.

otherWindowNumber

The number of the window the window is to be placed in front of or behind. Pass `0` to place the window in front of (when *orderingMode* is `NSWindowAbove`) or behind (when *orderingMode* is `NSWindowBelow`) all other windows in its level.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `orderFront:` (page 3351)
- `orderBack:` (page 3350)
- `orderOut:` (page 3352)
- `makeKeyAndOrderFront:` (page 3345)
- `level` (page 3343)
- `windowNumber` (page 3408)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSWindow.h`

parentWindow

Returns the parent window to which the window is attached as a child.

```
- (NSWindow *)parentWindow
```

Return Value
The window to which the window is attached as a child.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `removeChildWindow:` (page 3358)
- `childWindows` (page 3308)
- `addChildWindow:ordered:` (page 3297)
- `setParentWindow:` (page 3392)

Related Sample Code
FunkyOverlayWindow

Declared In
NSWindow.h

performClose:

This action method simulates the user clicking the close button by momentarily highlighting the button and then closing the window.

```
- (void)performClose:(id)sender
```

Parameters

sender
The message's sender.

Discussion
If the window's delegate or the window itself implements `windowShouldClose:` (page 3937), that message is sent with the window as the argument. (Only one such message is sent; if both the delegate and the `NSWindow` object implement the method, only the delegate receives the message.) If the `windowShouldClose:` (page 3937) method returns `NO`, the window isn't closed. If it returns `YES`, or if it isn't implemented, `performClose:` (page 3354) invokes the `close` (page 3308) method to close the window.

If the window doesn't have a close button or can't be closed (for example, if the delegate replies `NO` to a `windowShouldClose:` (page 3937) message), the system emits the alert sound.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `styleMask` (page 3402)
- `performMiniaturize:` (page 3354)

Related Sample Code
GLUT
QTMetadataEditor

Declared In
NSWindow.h

performMiniaturize:

Simulates the user clicking the minimize button by momentarily highlighting the button, then minimizing the window.

```
- (void)performMiniaturize:(id)sender
```

Parameters

sender
The message's sender.

Discussion
If the window doesn't have a minimize button or can't be minimized for some reason, the system emits the alert sound.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `close` (page 3308)
- `styleMask` (page 3402)
- `performClose:` (page 3354)

Declared In
`NSWindow.h`

performZoom:

This action method simulates the user clicking the zoom box by momentarily highlighting the button and then zooming the window.

```
- (void)performZoom:(id)sender
```

Parameters

sender
The object sending the message.

Discussion
If the window doesn't have a zoom box or can't be zoomed for some reason, the computer beeps.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `styleMask` (page 3402)
- `zoom:` (page 3409)

Declared In
`NSWindow.h`

postEvent:atStart:

Forwards the message to the global `NSApplication` object, `NSApp`.

```
- (void)postEvent:(NSEvent *)event atStart:(BOOL)atStart
```

Parameters

event
The event to add to the window's event queue.

atStart
`YES` to place the event in the front of the queue; `NO` to place it in the back.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `postEvent:atStart:` (page 157)

Declared In
NSWindow.h

preferredBackingLocation

Indicates the preferred location for the window's backing store.

```
- (NSWindowBackingLocation)preferredBackingLocation
```

Return Value
The preferred location for the window's backing store. See "Constants" (page 3411) for possible values.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setPreferredBackingLocation: (page 3393)
- backingLocation (page 3302)

Declared In
NSWindow.h

preservesContentDuringLiveResize

Returns whether the window tries to optimize user-initiated resize operations by preserving the content of views that have not changed.

```
- (BOOL)preservesContentDuringLiveResize
```

Return Value
YES if the window tries to optimize live resize operations by preserving the content of views that have not moved; otherwise, NO.

Discussion
When live-resize optimization is active, the window redraws only those views that moved (or do not support this optimization) during a live resize operation.

See preservesContentDuringLiveResize (page 3196) in NSView for additional information on how to support this optimization.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setPreservesContentDuringLiveResize: (page 3393)
- preservesContentDuringLiveResize (page 3196) (NSView)

Declared In
NSWindow.h

preventsApplicationTerminationWhenModal

Indicates whether the window prevents application termination when modal.

```
- (BOOL)preventsApplicationTerminationWhenModal
```

Return Value
`YES` if the window prevents application termination when modal; otherwise, `NO`.

Discussion
The default value is `YES`.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- setPreventsApplicationTerminationWhenModal:` (page 3394)

Declared In
`NSWindow.h`

print:

This action method runs the Print panel, and if the user chooses an option other than canceling, prints the window (its frame view and all subviews).

```
- (void)print:(id)sender
```

Parameters
sender
The message's sender.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

recalculateKeyViewLoop

Marks the key view loop as dirty and in need of recalculation.

```
- (void)recalculateKeyViewLoop
```

Discussion
The key view loop is actually recalculated the next time someone requests the next or previous key view of the window. The recalculated loop is based on the geometric order of the views in the window.

If you do not want to maintain the key view loop of your window manually, you can use this method to do it for you. When it is first loaded, `NSWindow` calls this method automatically if your window does not have a key view loop already established. If you add or remove views later, you can call this method manually to update the window's key view loop. You can also call `setAutorecalculatesKeyViewLoop:` (page 3370) to have the window recalculate the loop automatically.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `selectKeyViewFollowingView:` (page 3364)
- `selectKeyViewPrecedingView:` (page 3364)
- `setAutorecalculatesKeyViewLoop:` (page 3370)

Declared In
`NSWindow.h`

registerForDraggedTypes:

Registers a give set of pasteboard types as the pasteboard types the window will accept as the destination of an image-dragging session.

```
- (void)registerForDraggedTypes:(NSArray *)pasteboardTypes
```

Parameters

pasteboardTypes
An array of the pasteboard types the window will accept as the destination of an image-dragging session.

Discussion
Registering an `NSWindow` object for dragged types automatically makes it a candidate destination object for a dragging session. `NSWindow` has a default implementation for many of the methods in the `NSDraggingDestination` informal protocol. The default implementation forwards each message to the delegate if the delegate responds to the selector of the message. The messages forwarded this way are `draggingEntered:` (page 3655), `draggingUpdated:` (page 3656), `draggingExited:` (page 3655), `prepareForDragOperation:` (page 3657), `performDragOperation:` (page 3657), and `concludeDragOperation:` (page 3654).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `unregisterDraggedTypes` (page 3405)

Declared In
`NSWindow.h`

removeChildWindow:

Detaches a given child window from the window.

```
- (void)removeChildWindow:(NSWindow *)childWindow
```

Parameters

childWindow
The child window to detach.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `addChildWindow:ordered:` (page 3297)
- `childWindows` (page 3308)
- `parentWindow` (page 3353)
- `setParentWindow:` (page 3392)

Declared In
`NSWindow.h`

representedFilename

Returns the pathname of the file the window represents.

```
- (NSString *)representedFilename
```

Return Value
The path to the file of the window's represented file.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setRepresentedFilename:` (page 3395)

Declared In
`NSWindow.h`

representedURL

Provides the URL of the file the window represents.

```
- (NSURL *)representedURL
```

Return Value
The URL for the file the window represents.

Discussion
When the URL specifies a path, the window shows an icon in its title bar, as described in Table 170-1.

Table 170-1 Title bar document icon display

Filepath	Document icon
Empty	None.
Specifies a nonexistent file	Generic.
Specifies an existent file	Specific for the file's type.

You can customize the file icon in the tile bar with the following code:

```
[[<window> standardWindowButton:NSWindowDocumentIconButton] setImage:<image>]
```

When the URL identifies an existing file, the window's title offers a pop-up menu showing the path components of the URL. (The user displays this menu by Command-clicking the title.) The behavior and contents of this menu can be controlled with `window:shouldPopUpDocumentPathMenu:` (page 3931).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setRepresentedURL:` (page 3395)
- `window:shouldDragDocumentWithEvent:from:withPasteboard:` (page 3930) (`NSWindowDelegate`)

Declared In
`NSWindow.h`

resetCursorRects

Clears the window's cursor rectangles and the cursor rectangles of the `NSView` objects in its view hierarchy.

```
- (void)resetCursorRects
```

Discussion
Invokes `discardCursorRects` (page 3320) to clear the window's cursor rectangles, then sends `resetCursorRects` (page 3360) to every `NSView` object in the window's view hierarchy.

This method is typically invoked by the NSApplication object when it detects that the key window's cursor rectangles are invalid. In program code, it's more efficient to invoke `invalidateCursorRectsForView:` (page 3335).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
TextLinks

Declared In
`NSWindow.h`

resignKeyWindow

Invoked automatically when the window resigns key window status; never invoke this method directly.

```
- (void)resignKeyWindow
```

Discussion
This method sends `resignKeyWindow` (page 3360) to the window's first responder, sends `windowDidResignKey:` (page 3936) to the window's delegate, and posts an `NSWindowDidResignKeyNotification` (page 3425) to the default notification center.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `becomeKeyWindow` (page 3302)

- resignMainWindow (page 3361)

Declared In
NSWindow.h

resignMainWindow

Invoked automatically when the window resigns main window status; never invoke this method directly.

```
- (void)resignMainWindow
```

Discussion
This method sends windowDidResignMain: (page 3936) to the window's delegate and posts an NSWindowDidResignMainNotification (page 3425) to the default notification center.

Availability
Available in Mac OS X v10.0 and later.

See Also
- becomeMainWindow (page 3303)
- resignKeyWindow (page 3360)

Declared In
NSWindow.h

resizeFlags

Returns the flags field of the event record for the mouse-down event that initiated the resizing session.

```
- (NSInteger)resizeFlags
```

Return Value
A mask indicating which of the modifier keys was held down when the mouse-down event occurred. The flags are listed in NSEvent object's modifierFlags (page 1082) method description.

Discussion
This method is valid only while the window is being resized

You can use this method to constrain the direction or amount of resizing. Because of its limited validity, this method should only be invoked from within an implementation of the delegate method windowWillResize:toSize: (page 3940).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

resizeIncrements

Returns the window's resizing increments.

```
- (NSSize)resizeIncrements
```

Return Value
The window's resizing increments.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setResizeIncrements:` (page 3396)
- `setAspectRatio:` (page 3368)
- `setFrame:display:` (page 3381)

Declared In
`NSWindow.h`

restoreCachedImage

Splices the window's cached image rectangles, if any, back into its raster image (and buffer if it has one), undoing the effect of any drawing performed within those areas since they were established using `cacheImageInRect:` (page 3303).

```
- (void)restoreCachedImage
```

Discussion
You must invoke `flushWindow` (page 3327) after this method to guarantee proper redisplay. An `NSWindow` object automatically discards its cached image rectangles when it displays.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `discardCachedImage` (page 3320)
- `display` (page 3321)

Declared In
`NSWindow.h`

runToolbarCustomizationPalette:

The action method for the "Customize Toolbar…" menu item.

```
- (void)runToolbarCustomizationPalette:(id)sender
```

Parameters
sender
The message's sender.

Discussion
See the `NSToolbar` class description for additional information.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

saveFrameUsingName:

Saves the window's frame rectangle in the user defaults system under a given name.

```
- (void)saveFrameUsingName:(NSString *)frameName
```

Parameters

frameName
The name under which the frame is to be saved.

Discussion
With the companion method `setFrameUsingName:` (page 3384), you can save and reset an `NSWindow` object's frame over various launches of an application. The default is owned by the application and stored under the name `"NSWindow Frame frameName"`. See `NSUserDefaults` for more information.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- stringWithSavedFrame` (page 3402)

Declared In
NSWindow.h

screen

Returns the screen the window is on.

```
- (NSScreen *)screen
```

Return Value
The screen where most of the window is on; `nil` when the window is offscreen.

Discussion
When the window is partly on one screen and partly on another, the screen where most of it lies is returned.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- deepestScreen` (page 3315)

Related Sample Code
CocoaDVDPlayer
GLUT
iSpend
QTQuartzPlayer

Declared In
NSWindow.h

selectKeyViewFollowingView:

Makes key the view that follows the given view.

```
- (void)selectKeyViewFollowingView:(NSView *)referenceView
```

Parameters

referenceView

The view whose following view in the key view loop is sought.

Discussion

Sends the `nextValidKeyView` (page 3193) message to *referenceView* and, if that message returns an `NSView` object, invokes `makeFirstResponder:` (page 3344) with the returned object.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- selectKeyViewPrecedingView:` (page 3364)

Declared In

`NSWindow.h`

selectKeyViewPrecedingView:

Makes key the view that precedes the given view.

```
- (void)selectKeyViewPrecedingView:(NSView *)referenceView
```

Parameters

referenceView

The view whose preceding view in the key view loop is sought.

Discussion

Sends the `previousValidKeyView` (page 3197) message to *referenceView* and, if that message returns an `NSView` object, invokes `makeFirstResponder:` (page 3344) with the returned object.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- selectKeyViewFollowingView:` (page 3364)

Declared In

`NSWindow.h`

selectNextKeyView:

This action method searches for a candidate next key view and, if it finds one, invokes `makeFirstResponder:` (page 3344) to establish it as the first responder.

```
- (void)selectNextKeyView:(id)sender
```

Parameters

sender

The message's sender.

Discussion

The candidate is one of the following (searched for in this order):

- The current first responder's next valid key view, as returned by the `nextValidKeyView` (page 3193) method of `NSView`
- The object designated as the window's initial first responder (using `setInitialFirstResponder:` (page 3387)) if it returns `YES` to an `acceptsFirstResponder` (page 2143) message
- Otherwise, the initial first responder's next valid key view, which may end up being `nil`

Availability

Available in Mac OS X v10.0 and later.

See Also

`- selectPreviousKeyView:` (page 3365)

`- selectKeyViewFollowingView:` (page 3364)

Declared In

`NSWindow.h`

selectPreviousKeyView:

This action method searches for a candidate previous key view and, if it finds one, invokes `makeFirstResponder:` (page 3344) to establish it as the first responder.

```
- (void)selectPreviousKeyView:(id)sender
```

Parameters

sender

The message's sender.

Discussion

The candidate is one of the following (searched for in this order):

- The current first responder's previous valid key view, as returned by the `previousValidKeyView` (page 3197) method of `NSView`
- The object designated as the window's initial first responder (using `setInitialFirstResponder:` (page 3387)) if it returns `YES` to an `acceptsFirstResponder` (page 2143) message
- Otherwise, the initial first responder's previous valid key view, which may end up being `nil`

Availability

Available in Mac OS X v10.0 and later.

See Also

`- selectNextKeyView:` (page 3364)

`- selectKeyViewPrecedingView:` (page 3364)

Declared In
NSWindow.h

sendEvent:

This action method dispatches mouse and keyboard events sent to the window by the NSApplication object.

```
- (void)sendEvent:(NSEvent *)event
```

Parameters

event

The mouse or keyboard event to process.

Discussion
Never invoke this method directly. A right mouse-down event in a window of an inactive application is not delivered to the corresponding NSWindow object. It is instead delivered to the NSApplication object through a sendEvent: (page 167) message with a window number of 0.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

setAcceptsMouseMovedEvents:

Specifies whether the window is to accept mouse-moved events.

```
- (void)setAcceptsMouseMovedEvents:(BOOL)acceptMouseMovedEvents
```

Parameters

acceptMouseMovedEvents

YES to have the window accept mouse-moved events (and to distribute them to its responders); NO to not accept such events.

Availability
Available in Mac OS X v10.0 and later.

See Also
– acceptsMouseMovedEvents (page 3296)

Related Sample Code
GLUT

Declared In
NSWindow.h

setAllowsConcurrentViewDrawing:

Specifies whether the window allows its views to be drawn concurrently.

```
- (void)setAllowsConcurrentViewDrawing:(BOOL)flag
```

Parameters

flag

If YES, the window allows its views to be drawn concurrently; if NO, it does not.

Availability
Available in Mac OS X v10.6 and later.

See Also
- allowsConcurrentViewDrawing (page 3298)

Related Sample Code
DispatchFractal

Declared In
NSWindow.h

setAllowsToolTipsWhenApplicationIsInactive:

Specifies whether the window can display tooltips even when the application is in the background.

```
- (void)setAllowsToolTipsWhenApplicationIsInactive:(BOOL)allowTooltipsWhenAppInactive
```

Parameters

allowTooltipsWhenAppInactive

YES to have the window display tooltips even when its application is inactive; NO to suppress tooltip display when inactive.

Discussion
The message does not take effect until the window changes to an active state.

Note: Enabling tooltips in an inactive application will cause the application to do work any time the pointer passes over the window, thus degrading system performance.

Availability
Available in Mac OS X v10.3 and later.

See Also
- allowsToolTipsWhenApplicationIsInactive (page 3298)

Declared In
NSWindow.h

setAlphaValue:

Applies a given alpha value to the entire window.

```
- (void)setAlphaValue:(CGFloat)windowAlpha
```

Parameters

windowAlpha

The alpha value to apply.

Availability

Available in Mac OS X v10.0 and later.

See Also

- alphaValue (page 3298)

Related Sample Code

From A View to A Movie

FunkyOverlayWindow

MyMediaPlayer

UIElementInspector

Declared In

NSWindow.h

setAspectRatio:

Sets the window's aspect ratio, which constrains the size of its frame rectangle to integral multiples of this ratio when the user resizes it.

```
- (void)setAspectRatio:(NSSize)aspectRatio
```

Parameters

aspectRatio

The aspect ratio to be maintained during resizing actions.

Discussion

An NSWindow object's aspect ratio and its resize increments are mutually exclusive attributes. In fact, setting one attribute cancels the setting of the other. For example, to cancel an established aspect ratio setting for an NSWindow object, you send it a setResizeIncrements: (page 3396) message with the width and height set to 1.0:

```
[myWindow setResizeIncrements:NSMakeSize(1.0,1.0)];
```

The setContentAspectRatio: (page 3373) method takes precedence over this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- aspectRatio (page 3300)
- setFrame:display: (page 3381)

Declared In

NSWindow.h

setAutodisplay:

Specifies whether the window is to automatically display the views that are marked as needing it.

```
- (void)setAutodisplay:(BOOL)autodisplay
```

Parameters

autodisplay

If YES, the window will automatically display views that need to be displayed; if NO, it will not.

Discussion

If *autodisplay* is NO, the window or its views must be explicitly displayed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- isAutodisplay (page 3336)
- displayIfNeeded (page 3321)
- displayIfNeeded (page 3164) (NSView)

Declared In

NSWindow.h

setAutorecalculatesContentBorderThickness:forEdge:

Specifies whether the window calculates the thickness of a given border automatically.

```
- (void)setAutorecalculatesContentBorderThickness:(BOOL)flag forEdge:(NSRectEdge)edge
```

Parameters

autorecalculateContentBorderThickness

If YES, the window calculates the thickness of the edge automatically; if NO, it does not.

edge

The border whose thickness auto-recalculation status to set:

- NSMaxYEdge: Top border.
- NSMinYEdge: Bottom border.

Special Considerations

Turning off a border's auto-recalculation status sets its border thickness to 0.0.

In a non-textured window calling setAutorecalculatesContentBorderThickness:forEdge: passing NSMaxYEdge will raise an exception. It is only valid to set the content border thickness of the top edge in a textured window.

Availability

Available in Mac OS X v10.5 and later.

See Also

- autorecalculatesContentBorderThicknessForEdge: (page 3300)
- contentBorderThicknessForEdge: (page 3310)

Declared In

NSWindow.h

setAutorecalculatesKeyViewLoop:

Specifies whether to recalculate the key view loop automatically when views are added or removed.

```
- (void)setAutorecalculatesKeyViewLoop:(BOOL)autorecalculateKeyViewLoop
```

Parameters

autorecalculateKeyViewLoop

If YES,the window recalculates the key view loop automatically; if NO, it does not.

Discussion

If *autorecalculateKeyViewLoop* is NO, the client code must update the key view loop manually or call recalculateKeyViewLoop (page 3357) to have the window recalculate it.

Availability

Available in Mac OS X v10.4 and later.

See Also

- autorecalculatesKeyViewLoop (page 3301)
- recalculateKeyViewLoop (page 3357)

Declared In

NSWindow.h

setBackgroundColor:

Sets the window's background color to the given color.

```
- (void)setBackgroundColor:(NSColor *)color
```

Parameters

color

Color to set as the window's background color.

Availability

Available in Mac OS X v10.0 and later.

See Also

- backgroundColor (page 3301)

Related Sample Code

CIAnnotation

MyMediaPlayer

UIElementInspector

Declared In

NSWindow.h

setBackingType:

Sets the window's backing store type to a given type.

```
- (void)setBackingType:(NSBackingStoreType)backingType
```

Parameters

backingType

The backing store type to set.

Discussion

The valid backing store types are described in "Constants" (page 3411).

This method can be used only to switch a buffered window to retained or vice versa; you can't change the backing type to or from nonretained after initializing an `NSWindow` object (an error is generated if you attempt to do so).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `backingType` (page 3302)
- `initWithContentRect:styleMask:backing:defer:` (page 3332)
- `initWithContentRect:styleMask:backing:defer:screen:` (page 3333)

Declared In

`NSWindow.h`

setCanBecomeVisibleWithoutLogin:

Specifies whether the window can be displayed at the login window.

```
- (void)setCanBecomeVisibleWithoutLogin:(BOOL)flag
```

Parameters

flag

`YES` to allow the window to be displayed at the login window; `NO` to prevent this behavior.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `canBecomeVisibleWithoutLogin` (page 3305)

Related Sample Code

PreLoginAgents

Declared In

`NSWindow.h`

setCanBeVisibleOnAllSpaces:

Specifies whether the window can be visible on all spaces or on only one space at a time. (Deprecated in Mac OS X v10.5. Use `collectionBehavior` (page 3309) instead.)

```
- (void)setCanBeVisibleOnAllSpaces:(BOOL)flag
```

Parameters

flag

YES specifies that the window can be visible on all spaces; NO specifies that the window can be visible on only one space at a time.

Availability
Available in Mac OS X v10.5 and later.
Deprecated in Mac OS X v10.5.

Declared In
NSWindow.h

setCanHide:

Specifies whether the window can be hidden when its application becomes hidden (during execution of the NSApplication hide: (page 148) method).

```
- (void)setCanHide:(BOOL)canHide
```

Parameters

canHide

If YES, the window can be hidden when its application becomes hidden; if NO, it cannot.

Availability
Available in Mac OS X v10.0 and later.

See Also
- canHide (page 3306)

Declared In
NSWindow.h

setCollectionBehavior:

Specifies the window's behavior in window collections.

```
- (void)setCollectionBehavior:(NSWindowCollectionBehavior)behavior
```

Parameters

collectionBehavior

The collection behavior identifier to set.

Availability
Available in Mac OS X v10.5 and later.

See Also
- collectionBehavior (page 3309)

Declared In
NSWindow.h

setColorSpace:

Sets the window's color space.

```
- (void)setColorSpace:(NSColorSpace *)colorSpace
```

Parameters

colorSpace

The color space to set.

Availability
Available in Mac OS X v10.6 and later.

See Also
- colorSpace (page 3309)

Declared In
NSWindow.h

setContentAspectRatio:

Sets the aspect ratio (height in relation to width) of the window's content view, constraining the dimensions of its content rectangle to integral multiples of that ratio when the user resizes it.

```
- (void)setContentAspectRatio:(NSSize)contentAspectRatio
```

Parameters

contentAspectRatio

The aspect ratio of the window's content view.

Discussion
You can set a window's content view to any size programmatically, regardless of its aspect ratio. This method takes precedence over setAspectRatio: (page 3368).

Availability
Available in Mac OS X v10.3 and later.

See Also
- contentAspectRatio (page 3310)

Declared In
NSWindow.h

setContentBorderThickness:forEdge:

Specifies the thickness of a given border of the window.

```
- (void)setContentBorderThickness:(CGFloat)thickness forEdge:(NSRectEdge)edge
```

Parameters

borderThickness

The thickness for *edge*, in points.

edge

The border whose thickness to set:

- `NSMaxYEdge`: Top border.
- `NSMinYEdge`: Bottom border.

Discussion
In a non-textured window calling `setContentBorderThickness:forEdge:` passing `NSMaxYEdge` will raise an exception. It is only valid to set the content border thickness of the top edge in a textured window.

The `contentBorder` does not include the titlebar or toolbar, so a textured window that just wants the gradient in the titlebar and toolbar should have a `contentBorderThickness` of `0` for `NSMaxYEdge`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `contentBorderThicknessForEdge:` (page 3310)

Declared In
`NSWindow.h`

setContentMaxSize:

Sets the maximum size of the window's content view in the window's base coordinate system.

```
- (void)setContentMaxSize:(NSSize)contentMaxSize
```

Parameters
contentMaxSize

The maximum size of the window's content view in the window's base coordinate system.

Discussion
The maximum size constraint is enforced for resizing by the user as well as for the `setContentSize:` (page 3375) method and the `setFrame...` methods other than `setFrame:display:` (page 3381) and `setFrame:display:animate:` (page 3381). This method takes precedence over `setMaxSize:` (page 3388).

Availability
Available in Mac OS X v10.3 and later.

See Also
- `contentMaxSize` (page 3311)
- `setContentMinSize:` (page 3374)

Declared In
`NSWindow.h`

setContentMinSize:

Sets the minimum size of the window's content view in the window's base coordinate system.

```
- (void)setContentMinSize:(NSSize)contentMinSize
```

Parameters

contentMinSize

The minimum size of the window's content view in the window's base coordinate system.

Discussion

The minimum size constraint is enforced for resizing by the user as well as for the `setContentSize:` (page 3375) method and the `setFrame...` methods other than `setFrame:display:` (page 3381) and `setFrame:display:animate:` (page 3381). This method takes precedence over `setMinSize:` (page 3390).

Availability

Available in Mac OS X v10.3 and later.

See Also

- `contentMinSize` (page 3311)
- `setContentMaxSize:` (page 3374)

Declared In

`NSWindow.h`

setContentResizeIncrements:

Restricts the user's ability to resize the window so the width and height of its content view change by multiples of width and height increments.

```
- (void)setContentResizeIncrements:(NSSize)contentResizeIncrements
```

Parameters

contentResizeIncrements

The content-view resizing increments to set.

Discussion

As the user resizes the window, the size of its content view changes by integral multiples of *contentResizeIncrements*`.width` and *contentResizeIncrements*`.height`. However, you can set a window's size to any width and height programmatically. This method takes precedence over `setResizeIncrements:` (page 3396).

Availability

Available in Mac OS X v10.3 and later.

See Also

- `contentResizeIncrements` (page 3312)

Declared In

`NSWindow.h`

setContentSize:

Sets the size of the window's content view to a given size, which is expressed in the window's base coordinate system.

```
- (void)setContentSize:(NSSize)size
```

Parameters

size

The new size of the window's content view in the window's base coordinate system.

Discussion

This size in turn alters the size of the `NSWindow` object itself. Note that the window server limits window sizes to 10,000; if necessary, be sure to limit *aSize* relative to the frame rectangle.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setFrame:display:` (page 3381)

`+ contentRectForFrameRect:styleMask:` (page 3292)

`+ frameRectForContentRect:styleMask:` (page 3293)

Related Sample Code

CocoaVideoFrameToGWorld

QTAudioExtractionPanel

QTKitPlayer

Quartz Composer WWDC 2005 TextEdit

VideoViewer

Declared In

`NSWindow.h`

setContentView:

Makes a given view the window's content view.

```
- (void)setContentView:(NSView *)view
```

Parameters

view

View that is to become the window's content view.

Discussion

The window retains the new content view and owns it thereafter. The *view* object is resized to fit precisely within the content area of the window. You can modify the content view's coordinate system through its bounds rectangle, but can't alter its frame rectangle (that is, its size or location) directly.

This method causes the old content view to be released; if you plan to reuse it, be sure to retain it before sending this message and to release it as appropriate when adding it to another `NSWindow` object or `NSView`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- contentView` (page 3313)

`- setContentSize:` (page 3375)

Related Sample Code

CIFilterGeneratorTest

CustomSave
FunkyOverlayWindow
GLUT
OpenCL NBody Simulation Example

Declared In
NSWindow.h

setDefaultButtonCell:

Makes the key equivalent of button cell the Return (or Enter) key, so when the user presses Return that button performs as if clicked.

```
- (void)setDefaultButtonCell:(NSButtonCell *)defaultButtonCell
```

Parameters

defaultButtonCell
The button cell to perform as if clicked when the window receives a Return (or Enter) key event.

Availability
Available in Mac OS X v10.0 and later.

See Also
- defaultButtonCell (page 3316)
- disableKeyEquivalentForDefaultButtonCell (page 3319)
- enableKeyEquivalentForDefaultButtonCell (page 3324)

Related Sample Code
BackgroundExporter

Declared In
NSWindow.h

setDelegate:

Sets the window's delegate to a given object or removes an existing delegate.

```
- (void)setDelegate:(id < NSWindowDelegate >)delegate
```

Parameters

delegate
The delegate for the window. Pass nil to remove an existing delegate.

Discussion
An NSWindow object's delegate is inserted in the responder chain after the window itself and is informed of various actions by the window through delegation messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
- delegate (page 3316)

- tryToPerform:with: (page 3404)
- sendAction:to:from: (page 166) (NSApplication)

Related Sample Code
AudioBurn
DataBurn
Eraser
Quartz Composer WWDC 2005 TextEdit
Verification

Declared In
NSWindow.h

setDepthLimit:

Sets the depth limit of the window to a given limit.

```
- (void)setDepthLimit:(NSWindowDepth)depthLimit
```

Parameters
depthLimit
The depth limit to set.

Discussion
The NSBestDepth (page 3965) function provides the best depth limit based on a set of parameters.

Passing a value of 0 for *depthLimit* sets the depth limit to the window's default depth limit. A depth limit of 0 can be useful for reverting an NSWindow object to its initial depth.

On Mac OS X 10.6 and later, you can pass one of the explicit bit depths defined in "Explicit Window Depth Limits" (page 3417)NSWindowDepthTwentyfourBitRGB (page 3417) is the default.

Availability
Available in Mac OS X v10.0 and later.

See Also
- depthLimit (page 3317)
+ defaultDepthLimit (page 3292)
- setDynamicDepthLimit: (page 3380)

Declared In
NSWindow.h

setDisplaysWhenScreenProfileChanges:

Specifies whether the window context should be updated when the screen profile changes.

```
- (void)setDisplaysWhenScreenProfileChanges:(BOOL)displaysWhenScreenProfileChanges
```

Parameters

displaysWhenScreenProfileChanges

- YES specifies that the window context should be changed in these situations:
  - A majority of the window is moved to a different screen whose profile is different than the previous screen.
  - The ColorSync profile of the current screen changes.
- NO specifies that the screen profile information for the window context doesn't change.

Discussion

After the window context is updated, the window is told to display itself. If you need to update offscreen caches for the window, you should register to receive the NSWindowDidChangeScreenProfileNotification (page 3423) notification.

Availability

Available in Mac OS X v10.4 and later.

See Also

- displaysWhenScreenProfileChanges (page 3322)

Related Sample Code

ImageApp

Declared In

NSWindow.h

setDocumentEdited:

Specifies whether the window's document has been edited.

```
- (void)setDocumentEdited:(BOOL)documentEdited
```

Parameters

documentEdited

If YES, the window's document is marked as having been edited; if NO, it is marked as not having been edited.

Discussion

You should send setDocumentEdited:YES to an NSWindow object every time the window's document changes in such a way that it needs to be saved. Conversely, when the document is saved, you should send setDocumentEdited:NO. Then, before closing the window you can use isDocumentEdited (page 3336) to determine whether to allow the user a chance to save the document.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

GLUT

PDF Annotation Editor

PDFKitLinker2

Quartz Composer WWDC 2005 TextEdit

Declared In
NSWindow.h

setDynamicDepthLimit:

Sets whether the window changes its depth to match the depth of the screen it's on, or the depth of the deepest screen when it spans multiple screens.

```
- (void)setDynamicDepthLimit:(BOOL)dynamicDepthLimit
```

Parameters

dynamicDepthLimit

If YES, the window has a dynamic depth limit; if NO, it does not.

Discussion
When *dynamicDepthLimit* is NO, the window uses either its preset depth limit or the default depth limit. A different, and nondynamic, depth limit can be set with the setDepthLimit: (page 3378) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- hasDynamicDepthLimit (page 3330)
+ defaultDepthLimit (page 3292)

Declared In
NSWindow.h

setExcludedFromWindowsMenu:

Specifies whether the window's title is omitted from the application's Windows menu.

```
- (void)setExcludedFromWindowsMenu:(BOOL)excludedFromWindowsMenu
```

Parameters

excludedFromWindowsMenu

If YES, the window will be omitted from the application's Windows menu; if NO, it will not be omitted.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isExcludedFromWindowsMenu (page 3337)

Related Sample Code
From A View to A Movie
From A View to A Picture
VertexPerformanceTest

Declared In
NSWindow.h

setFrame:display:

Sets the origin and size of the window's frame rectangle according to a given frame rectangle, thereby setting its position and size onscreen.

```
- (void)setFrame:(NSRect)windowFrame display:(BOOL)displayViews
```

Parameters

windowFrame

The frame rectangle for the window, including the title bar.

displayViews

Specifies whether the window redraws the views that need to be displayed. When `YES` the window sends a `displayIfNeeded` (page 3321) message down its view hierarchy, thus redrawing all views.

Discussion

Note that the window server limits window position coordinates to ±16,000 and sizes to 10,000.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `frame` (page 3328)
- `setFrameFromString:` (page 3383)
- `setFrameOrigin:` (page 3383)
- `setFrameTopLeftPoint:` (page 3384)
- `setFrameUsingName:` (page 3384)
+ `frameRectForContentRect:styleMask:` (page 3293)

Related Sample Code

AnimatedTableView

BasicCocoaAnimations

CocoaCreateMovie

FunkyOverlayWindow

StickiesWithCoreData

Declared In

`NSWindow.h`

setFrame:display:animate:

Sets the origin and size of the window's frame rectangle, with optional animation, according to a given frame rectangle, thereby setting its position and size onscreen.

```
- (void)setFrame:(NSRect)windowFrame display:(BOOL)displayViews
    animate:(BOOL)performAnimation
```

Parameters

windowFrame

The frame rectangle for the window, including the title bar.

displayViews

Specifies whether the window redraws the views that need to be displayed. When `YES` the window sends a `displayIfNeeded` (page 3321) message down its view hierarchy, thus redrawing all views.

performAnimation

Specifies whether the window performs a smooth resize. `YES` to perform the animation, whose duration is specified by `animationResizeTime:` (page 3299).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- frame` (page 3328)
`+ frameRectForContentRect:styleMask:` (page 3293)

Related Sample Code
CIVideoDemoGL
DatePicker
FunHouse
iSpend
PredicateEditorSample

Declared In
`NSWindow.h`

setFrameAutosaveName:

Sets the name used to automatically save the window's frame rectangle in the defaults system to a given name.

```
- (BOOL)setFrameAutosaveName:(NSString *)frameName
```

Parameters

frameName

The name under which the frame is to be saved.

Return Value
`YES` when the frame name is set successfully; `NO` when frameName is being used as an autosave name by another `NSWindow` object in the application (in which case the window's old name remains in effect).

Discussion
If *frameName* isn't the empty string (@""), the window's frame is saved as a user default (as described in `saveFrameUsingName:` (page 3363)) each time the frame changes.

When the window has an autosave name, its frame data is written whenever the frame rectangle changes.

If there is a frame rectangle previously stored for *frameName* in the user defaults, the window's frame is set to this frame rectangle. That is, when you call this method with a previously used *frameName*, the window picks up the previously saved setting. For example, if you call `setFrameAutosaveName:` for a window that is already onscreen, this method could cause the window to move to a different screen location. For this reason, it is generally better to call this method before the window is visible on screen.

Keep in mind that a window controller may change the window's position when it displays it if window cascading is turned on. To preclude the window controller from changing a window's position from the one saved in the defaults system, you must send `setShouldCascadeWindows:NO` to the window controller.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ removeFrameUsingName: (page 3294)
- stringWithSavedFrame (page 3402)
- setFrameFromString: (page 3383)

Related Sample Code
CIRAWFilterSample

Declared In
NSWindow.h

setFrameFromString:

Sets the window's frame rectangle from a given string representation.

```
- (void)setFrameFromString:(NSString *)frameString
```

Parameters
frameString
A string representation of a frame rectangle, previously creating using stringWithSavedFrame (page 3402).

Discussion
If the window is not resizable, this method will not resize the window. The frame is constrained according to the window's minimum and maximum size settings. This method causes a windowWillResize:toSize: (page 3940) message to be sent to the delegate.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

setFrameOrigin:

Positions the bottom-left corner of the window's frame rectangle at a given point in screen coordinates.

```
- (void)setFrameOrigin:(NSPoint)point
```

Parameters
point
The new position of the window's bottom-left corner in screen coordinates.

Discussion
Note that the window server limits window position coordinates to ±16,000.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setFrame:display: (page 3381)
- setFrameTopLeftPoint: (page 3384)

Related Sample Code
BoingX
FunkyOverlayWindow
GLUT
MyMediaPlayer
StickiesWithCoreData

Declared In
NSWindow.h

setFrameTopLeftPoint:

Positions the top-left corner of the window's frame rectangle at a given point in screen coordinates.

```
- (void)setFrameTopLeftPoint:(NSPoint)point
```

Parameters
point
The new position of the window's top-left corner in screen coordinates.

Discussion
Note that the window server limits window position coordinates to ±16,000; if necessary, adjust *aPoint* relative to the window's lower-left corner to account for this limit.

Availability
Available in Mac OS X v10.0 and later.

See Also
- cascadeTopLeftFromPoint: (page 3306)
- setFrame:display: (page 3381)
- setFrameOrigin: (page 3383)

Related Sample Code
CocoaCreateMovie
CocoaDVDPlayer
DemoMonkey
QTAudioContextInsert
Quartz Composer WWDC 2005 TextEdit

Declared In
NSWindow.h

setFrameUsingName:

Sets the window's frame rectangle by reading the rectangle data stored under a given name from the defaults system.

```
- (BOOL)setFrameUsingName:(NSString *)frameName
```

Parameters

frameName

The name of the frame to read.

Return Value

YES when *frameName* is read and the frame is set successfully; otherwise, NO.

Discussion

The frame is constrained according to the window's minimum and maximum size settings. This method causes a windowWillResize:toSize: (page 3940) message to be sent to the delegate.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setFrameAutosaveName: (page 3382)
+ removeFrameUsingName: (page 3294)
- stringWithSavedFrame (page 3402)
- setFrameFromString: (page 3383)

Declared In

NSWindow.h

setFrameUsingName:force:

Sets the window's frame rectangle by reading the rectangle data stored under a given name from the defaults system. Can operate on nonresizable windows.

```
- (BOOL)setFrameUsingName:(NSString *)frameName force:(BOOL)force
```

Parameters

frameName

The name of the frame to read.

force

YES to use setFrameUsingName: (page 3384) on a nonresizable window; NO to fail on a nonresizable window.

Return Value

YES when *frameName* is read and the frame is set successfully; otherwise,NO.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSWindow.h

setHasShadow:

Specifies whether the window has a shadow.

```
- (void)setHasShadow:(BOOL)hasShadow
```

Parameters

hasShadow

If YES, the window has a shadow; if NO, it does not.

Discussion

If the shadow setting changes, the window shadow is invalidated, forcing the window shadow to be recomputed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- hasShadow (page 3331)
- invalidateShadow (page 3336)

Related Sample Code

FunkyOverlayWindow

Declared In

NSWindow.h

setHidesOnDeactivate:

Specifies whether the window is removed from the screen when the application is inactive.

```
- (void)setHidesOnDeactivate:(BOOL)hideOnDeactivate
```

Parameters

hideOnDeactivate

- YES specifies that the window is to be hidden (taken out of the screen list) when the application stops being the active application
- NO specifies that the window is to remain onscreen when the application becomes inactive.

Availability

Available in Mac OS X v10.0 and later.

See Also

- hidesOnDeactivate (page 3331)

Related Sample Code

From A View to A Movie

From A View to A Picture

GLFullScreen

PreLoginAgents

Declared In

NSWindow.h

setIgnoresMouseEvents:

Specifies whether the window is transparent to mouse clicks and other mouse events, allowing overlay windows.

```
- (void)setIgnoresMouseEvents:(BOOL)ignoreMouseEvents
```

Parameters

ignoreMouseEvents

If YES, the window will ignore mouse events; if NO, it will not.

Availability
Available in Mac OS X v10.2 and later.

See Also
- ignoresMouseEvents (page 3331)

Related Sample Code
FunkyOverlayWindow
UIElementInspector

Declared In
NSWindow.h

setInitialFirstResponder:

Sets a given view as the one that's made first responder (also called the key view) the first time the window is placed onscreen.

```
- (void)setInitialFirstResponder:(NSView *)view
```

Parameters

view

The view to make first responder the first time the window is placed onscreen.

Availability
Available in Mac OS X v10.0 and later.

See Also
- initialFirstResponder (page 3332)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
NSWindow.h

setLevel:

Sets the window's window level to a given level.

```
- (void)setLevel:(NSInteger)windowLevel
```

Parameters

windowLevel

The window level to set.

Discussion

Some useful predefined values, ordered from lowest to highest, are described in "Window Levels" (page 3412).

Each level in the list groups windows within it in front of those in all preceding groups. Floating windows, for example, appear in front of all normal-level windows. When a window enters a new level, it's ordered in front of all its peers in that level.

The constant `NSTornOffMenuWindowLevel` is preferable to its synonym, `NSSubmenuWindowLevel`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `level` (page 3343)
- `orderWindow:relativeTo:` (page 3352)
- `orderFront:` (page 3351)
- `orderBack:` (page 3350)

Related Sample Code

FunkyOverlayWindow
GLFullScreen
GLUT
Quartz Composer WWDC 2005 TextEdit
UIElementInspector

Declared In

`NSWindow.h`

setMaxSize:

Sets the maximum size to which the window's frame (including its title bar) can be sized.

```
- (void)setMaxSize:(NSSize)maxFrameSize
```

Parameters

maxFrameSize

The maximum size of the window's frame.

Discussion

The maximum size constraint is enforced for resizing by the user as well as for the `setFrame...` methods other than `setFrame:display:` (page 3381) and `setFrame:display:animate:` (page 3381). Note that the window server limits window sizes to 10,000.

The default maximum size of a window is `{FLT_MAX, FLT_MAX}` (`FLT_MAX` is defined in `/usr/include/float.h`). Once the maximum size of a window has been set, there is no way to reset it other than specifying this default maximum size.

The setContentMaxSize: (page 3374) method takes precedence over this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `maxSize` (page 3346)
- `setMinSize:` (page 3390)
- `setAspectRatio:` (page 3368)
- `setResizeIncrements:` (page 3396)

Related Sample Code
CIAnnotation

Declared In
`NSWindow.h`

setMiniwindowImage:

Sets the window's custom minimized window image to a given image.

```
- (void)setMiniwindowImage:(NSImage *)miniwindowImage
```

Parameters

miniwindowImage
Image to set as the window's minimized window image.

Discussion
When the user minimizes the window, the Dock displays *miniwindowImage* in the corresponding Dock tile, scaling it as needed to fit in the tile. If you do not specify a custom image using this method, the Dock creates one for you automatically.

You can also call this method as needed to change the minimized window image. Typically, you would specify a custom image immediately prior to a window being minimized—when the system posts an `NSWindowWillMiniaturizeNotification` (page 3427). You can call this method while the window is minimized to update the current image in the Dock. However, this method is not recommended for creating complex animations in the Dock.

Support for custom images is disabled by default. To enable support, set the `AppleDockIconEnabled` key to `YES` when first registering your application's user defaults. You must set this key prior to calling the `init` method of `NSApplication`, which reads the current value of the key.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `miniwindowImage` (page 3347)
- `isMiniaturized` (page 3338)

Declared In
`NSWindow.h`

setMiniwindowTitle:

Sets the title of the window's miniaturized counterpart to a given string and redisplays it.

```
- (void)setMiniwindowTitle:(NSString *)miniwindowTitle
```

Parameters

miniwindowTitle

The string to set as the title of the minimized window.

Discussion

A minimized window's title normally reflects that of its full-size counterpart, abbreviated to fit if necessary. Although this method allows you to set the minimized window's title explicitly, changing the full-size `NSWindow` object's title (through `setTitle:` (page 3398) or `setTitleWithRepresentedFilename:` (page 3399)) automatically changes the minimized window's title as well.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `miniwindowTitle` (page 3347)

Related Sample Code

GLUT

Declared In

`NSWindow.h`

setMinSize:

Sets the minimum size to which the window's frame (including its title bar) can be sized to *aSize*.

```
- (void)setMinSize:(NSSize)minFrameSize
```

Parameters

minFrameSize

The minimum size of the window's frame.

Discussion

The minimum size constraint is enforced for resizing by the user as well as for the `setFrame...` methods other than `setFrame:display:` (page 3381) and `setFrame:display:animate:` (page 3381).

The `setContentMinSize:` (page 3374) method takes precedence over this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `minSize` (page 3348)
- `setMaxSize:` (page 3388)
- `setAspectRatio:` (page 3368)
- `setResizeIncrements:` (page 3396)

Declared In

`NSWindow.h`

setMovable:

Specifies whether the window can be dragged by clicking in its title bar or background.

```
- (void)setMovable:(BOOL)flag
```

Parameters

flag

If YES, dragging is enabled; if NO, it is disabled.

Availability
Available in Mac OS X v10.6 and later.

See Also
- isMovable (page 3339)

Declared In
NSWindow.h

setMovableByWindowBackground:

Sets whether the window is movable by clicking and dragging anywhere in its background.

```
- (void)setMovableByWindowBackground:(BOOL)movableByWindowBackground
```

Parameters

movableByWindowBackground

YES to specify that the window is movable by background, NO to specify that the window is not movable by background.

Availability
Available in Mac OS X v10.2 and later.

See Also
- isMovableByWindowBackground (page 3339)

Related Sample Code
MyMediaPlayer

Declared In
NSWindow.h

setOneShot:

Sets whether the window device that the window manages should be freed when it's removed from the screen list.

```
- (void)setOneShot:(BOOL)oneShot
```

Parameters

oneShot

YES to free the window's window device when it's removed from the screen list (hidden) and to create another one when it's returned to the screen; NO to reuse the window device.

Discussion
Freeing the window device when it's removed from the screen list can result in memory savings and performance improvement for `NSWindow` objects that don't take long to display. It's particularly appropriate for `NSWindow` objects the user might use once or twice but not display continually.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- isOneShot` (page 3340)

Related Sample Code
CIBevelSample
CIFilterGeneratorTest
CIMicroPaint
FunHouse
VideoViewer

Declared In
`NSWindow.h`

setOpaque:

Specifies whether the window is opaque.

```
- (void)setOpaque:(BOOL)opaque
```

Parameters

opaque
If `YES`, the window is opaque; if `NO`, it is not.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- isOpaque` (page 3340)

Related Sample Code
CIAnnotation
FunkyOverlayWindow
GLFullScreen
UIElementInspector

Declared In
`NSWindow.h`

setParentWindow:

Adds the window as a child of a given window. For use by subclasses when setting the parent window in the window.

```
- (void)setParentWindow:(NSWindow *)parentWindow
```

Parameters

parentWindow

The window to be a child of the given window.

Discussion

This method should be called from a subclass when it is overridden by a subclass's implementation. It should not be called otherwise.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `removeChildWindow:` (page 3358)
- `childWindows` (page 3308)
- `parentWindow` (page 3353)
- `addChildWindow:ordered:` (page 3297)

Related Sample Code

CIAnnotation

Declared In

`NSWindow.h`

setPreferredBackingLocation:

Specifies the preferred location for the window's backing store.

```
- (void)setPreferredBackingLocation:(NSWindowBackingLocation)backingLocation
```

Parameters

preferredBackingLocation

The preferred location for the window's backing store. See "NSWindowBackingLocation" (page 3420) for possible values.

Discussion

Use only when optimizing for performance.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `preferredBackingLocation` (page 3356)

Related Sample Code

DispatchFractal

Declared In

`NSWindow.h`

setPreservesContentDuringLiveResize:

Specifies whether the window tries to optimize live resize operations by preserving the content of views that have not changed.

```
- (void)setPreservesContentDuringLiveResize:(BOOL)preservesContentDuringLiveResize
```

Parameters

preservesContentDuringLiveResize

YES turns on live-resize optimization; NO turns it off for the window and all of its contained views.

Discussion

By default, live-resize optimization is turned on.

You might consider disabling this optimization for the window if none of the window's contained views can take advantage of it. Disabling the optimization for the window prevents it from checking each view to see if the optimization is supported.

Availability

Available in Mac OS X v10.4 and later.

See Also

- preservesContentDuringLiveResize (page 3356)

Declared In

NSWindow.h

setPreventsApplicationTerminationWhenModal:

Specifies whether the window prevents application termination when modal.

```
- (void)setPreventsApplicationTerminationWhenModal:(BOOL)flag
```

Parameters

flag

If YES, the window will prevent application termination when modal; if NO, it will not.

Discussion

Normally, application termination is prevented when a modal window or sheet is open, without consulting the application delegate. Some windows may wish not to prevent termination, however. Calling this method with an argument of NO overrides the default behavior and allows termination to proceed even if the window is open, either through the sudden termination path if enabled, or after consulting the application delegate.

Availability

Available in Mac OS X v10.6 and later.

See Also

- preventsApplicationTerminationWhenModal (page 3357)

Declared In

NSWindow.h

setReleasedWhenClosed:

Specifies whether the window is released when it receives the close message.

```
- (void)setReleasedWhenClosed:(BOOL)releasedWhenClosed
```

Parameters

releasedWhenClosed

YES to specify that the window is to be hidden and released when it receives a close message; NO to specify that the window is only hidden, not released.

Discussion

Another strategy for releasing an NSWindow object is to have its delegate autorelease it on receiving a windowShouldClose: (page 3937) message.

Availability

Available in Mac OS X v10.0 and later.

See Also

- close (page 3308)
- isReleasedWhenClosed (page 3341)

Related Sample Code

CIAnnotation
Fiendishthngs
From A View to A Movie
GLUT
WhackedTV

Declared In

NSWindow.h

setRepresentedFilename:

Sets the pathname of the file the window represents.

```
- (void)setRepresentedFilename:(NSString *)filePath
```

Parameters

filePath

The path to the file to set as the window's represented file.

Availability

Available in Mac OS X v10.0 and later.

See Also

- representedFilename (page 3359)
- setTitleWithRepresentedFilename: (page 3399)

Declared In

NSWindow.h

setRepresentedURL:

Specifies the URL of the file the window represents.

```
- (void)setRepresentedURL:(NSURL *)url
```

Parameters

representedURL

The URL of the file the window is to represent.

Availability

Available in Mac OS X v10.5 and later.

See Also

- representedURL (page 3359)

Declared In

NSWindow.h

setResizeIncrements:

Restricts the user's ability to resize the window so the width and height change by multiples of width and height increments.

```
- (void)setResizeIncrements:(NSSize)resizeIncrements
```

Parameters

resizeIncrements

The resizing increments to set.

Discussion

As the user resizes the window, its size changes by multiples of *increments*.width and *increments*.height, which should be whole numbers, 1.0 or greater. Whatever the current resizing increments, you can set an NSWindow object's size to any height and width programmatically.

Resize increments and aspect ratio are mutually exclusive attributes. For more information, see setAspectRatio: (page 3368).

The setContentResizeIncrements: (page 3375) method takes precedence over this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- resizeIncrements (page 3361)
- setFrame:display: (page 3381)

Declared In

NSWindow.h

setSharingType:

Specifies the level of access other processes have to the window's content.

```
- (void)setSharingType:(NSWindowSharingType)type
```

Parameters

sharingType

The sharing level of the window's content. See "NSWindowSharingType" (page 3419) for possible values.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `sharingType` (page 3401)

Declared In
`NSWindow.h`

setShowsResizeIndicator:

Specifies whether the window's resize indicator is visible

```
- (void)setShowsResizeIndicator:(BOOL)showResizeIndicator
```

Parameters

showResizeIndicator
Specifies the resize indicator state. `YES` to show it, `NO` to hide it.

Discussion
This method does not affect whether the window is resizable.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `showsResizeIndicator` (page 3401)

Related Sample Code
QTKitPlayer

Declared In
`NSWindow.h`

setShowsToolbarButton:

Specifies whether the window shows the toolbar control button.

```
- (void)setShowsToolbarButton:(BOOL)showsToolbarButton
```

Parameters

showsToolbarButton
`YES` to display the toolbar control button; `NO` to hide the button.

Discussion
If the window does not have a toolbar, this method has no effect.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `showsToolbarButton` (page 3401)

Declared In
NSWindow.h

setStyleMask:

Sets the window's style mask to the given value.

```
- (void)setStyleMask:(NSUInteger)styleMask
```

Parameters

styleMask
The new style mask value.

Discussion
The valid style mask values are the same values acceptable for use in `initWithContentRect:styleMask:backing:defer:` (page 3332). Some style mask changes cause the view hierarchy to be rebuilt.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- styleMask` (page 3402)

Related Sample Code
MyMediaPlayer

Declared In
NSWindow.h

setTitle:

Sets the string that appears in the window's title bar (if it has one) to a given string and displays the title.

```
- (void)setTitle:(NSString *)title
```

Parameters

title
The string to set as the window's title.

Discussion
Also sets the title of the window's miniaturized window.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- title` (page 3403)
`- setTitleWithRepresentedFilename:` (page 3399)
`- setMiniwindowTitle:` (page 3390)

Related Sample Code
BackgroundExporter

GLUT
OpenCL NBody Simulation Example
UIElementInspector
WhackedTV

Declared In
NSWindow.h

setTitleWithRepresentedFilename:

Sets a given path as the window's title, formatting it as a file-system path, and records this path as the window's associated filename using `setRepresentedFilename:` (page 3395).

```
- (void)setTitleWithRepresentedFilename:(NSString *)filePath
```

Parameters
filePath
The file path to set as the window's title.

Discussion
The filename—not the pathname—is displayed in the window's title bar.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `title` (page 3403)
- `setTitle:` (page 3398)
- `setMiniwindowTitle:` (page 3390)

Declared In
NSWindow.h

setToolbar:

Sets the window's toolbar.

```
- (void)setToolbar:(NSToolbar *)toolbar
```

Parameters
toolbar
The toolbar for the window.

Discussion
See the `NSToolbar` class description for additional information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `toolbar` (page 3404)

Related Sample Code
EnhancedDataBurn
iSpend
PDFKitLinker2
QTAudioExtractionPanel
QTKitPlayer

Declared In
NSWindow.h

setViewsNeedDisplay:

Specifies whether the window's views need to be displayed..

```
- (void)setViewsNeedDisplay:(BOOL)viewsNeedDisplay
```

Parameters
viewsNeedDisplay
If YES, the window's views are set to need to be displayed; if NO, they are not.

Discussion
You should rarely need to invoke this method; the NSView method setNeedsDisplay: (page 3225) and similar methods invoke it automatically.

Availability
Available in Mac OS X v10.0 and later.

See Also
- viewsNeedDisplay (page 3407)

Declared In
NSWindow.h

setWindowController:

Sets the window's window controller.

```
- (void)setWindowController:(NSWindowController *)windowController
```

Parameters
windowController
Window controller to set.

Availability
Available in Mac OS X v10.0 and later.

See Also
- windowController (page 3407)

Declared In
NSWindow.h

sharingType

Indicates the level of access other processes have to the window's content.

```
- (NSWindowSharingType)sharingType
```

Return Value
The sharing level of the window's content. See "NSWindowSharingType" (page 3419) for possible values.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setSharingType: (page 3396)

Declared In
NSWindow.h

showsResizeIndicator

Returns a Boolean value that indicates whether the window's resize indicator is visible.

```
- (BOOL)showsResizeIndicator
```

Return Value
YES when the window's resize indicator is visible; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setShowsResizeIndicator: (page 3397)

Declared In
NSWindow.h

showsToolbarButton

Indicates whether the toolbar control button is currently displayed.

```
- (BOOL)showsToolbarButton
```

Return Value
YES if the standard toolbar button is currently displayed; otherwise, NO.

Discussion
When clicked, the toolbar control button shows or hides a window's toolbar. The toolbar control button appears in a window's title bar.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setShowsToolbarButton: (page 3397)

Declared In
NSWindow.h

standardWindowButton:

Returns the window button of a given window button kind in the window's view hierarchy.

```
- (NSButton *)standardWindowButton:(NSWindowButton)windowButtonKind
```

Parameters

windowButtonKind
The kind of standard window button to return.

Return Value
Window button in the window's view hierarchy of the kind identified by *windowButtonKind*; `nil` when such button is not in the window's view hierarchy.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ standardWindowButton:forStyleMask:` (page 3295)

Related Sample Code
GLUT

Declared In
NSWindow.h

stringWithSavedFrame

Returns a string representation of the window's frame rectangle.

```
- (NSString *)stringWithSavedFrame
```

Return Value
A string representation of the window's frame rectangle in a format that can be used with a later `setFrameFromString:` (page 3383) message.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

styleMask

Returns the window's style mask, indicating what kinds of control items it displays.

```
- (NSUInteger)styleMask
```

Return Value
The window's style mask.

Discussion
See the information about the style mask in "Window Style Masks" (page 3411).

Availability
Available in Mac OS X v10.0 and later.

See Also
- setStyleMask: (page 3398)

Related Sample Code
CocoaDragAndDrop
GLUT
Quartz Composer WWDC 2005 TextEdit
StickiesWithCoreData

Declared In
NSWindow.h

title

Returns either the string that appears in the title bar of the window, or the path to the represented file.

```
- (NSString *)title
```

Return Value
The window's title or the path to the represented file.

Discussion
If the title has been set using `setTitleWithRepresentedFilename:` (page 3399), this method returns the file's path.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setTitle: (page 3398)

Related Sample Code
QTAudioExtractionPanel

Declared In
NSWindow.h

toggleToolbarShown:

The action method for the "Hide Toolbar" menu item (which alternates with "Show Toolbar").

```
- (void)toggleToolbarShown:(id)sender
```

Parameters
sender
The message's sender.

Discussion
See the `NSToolbar` class description for additional information.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

toolbar

Returns the window's toolbar.

```
- (NSToolbar *)toolbar
```

Return Value
The window's toolbar.

Discussion
See the `NSToolbar` class description for additional information.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setToolbar:` (page 3399)

Declared In
`NSWindow.h`

tryToPerform:with:

Dispatches action messages with a given argument.

```
- (BOOL)tryToPerform:(SEL)selector with:(id)object
```

Parameters

selector
The selector to attempt to execute.

object
The message's argument.

Return Value
`YES` when the window or its delegate perform *selector* with *object*; otherwise, `NO`.

Discussion
The window tries to perform the method *selector* using its inherited `NSResponder` method `tryToPerform:with:` (page 2202). If the window doesn't perform *selector*, the delegate is given the opportunity to perform it using its inherited `NSObject` method `performSelector:withObject:`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

unregisterDraggedTypes

Unregisters the window as a possible destination for dragging operations.

```
- (void)unregisterDraggedTypes
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- registerForDraggedTypes: (page 3358)

Declared In
NSWindow.h

update

Updates the window.

```
- (void)update
```

Discussion
The NSWindow implementation of this method does nothing more than post an NSWindowDidUpdateNotification (page 3426) notification to the default notification center. A subclass can override this method to perform specialized operations, but it should send an update message to super just before returning. For example, the NSMenu class implements this method to disable and enable menu commands.

An NSWindow object is automatically sent an update message on every pass through the event loop and before it's displayed onscreen. You can manually cause an update message to be sent to all visible NSWindow objects through the NSApplication updateWindows (page 179) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setWindowsNeedUpdate: (page 172) (NSApplication)

Related Sample Code
FunHouse

Declared In
NSWindow.h

useOptimizedDrawing:

Specifies whether the window is to optimize focusing and drawing when displaying its views.

```
- (void)useOptimizedDrawing:(BOOL)optimizedDrawing
```

Parameters

optimizedDrawing

If YES, the window will optimize focusing and drawing for its views; if NO, it will not, in which case, the window does not preserve the Z-ordering of overlapping views when an object explicitly sends lockFocus (page 3187) to a view and draws directly to it, instead of using the AppKit standard display mechanism.

Discussion

The optimizations may prevent sibling subviews from being displayed in the correct order—which matters only if the subviews overlap. You should always set *optimizedDrawing* to YES when there are no overlapping subviews within the window. The default is NO.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

DispatchFractal

Declared In

NSWindow.h

userSpaceScaleFactor

Returns the scale factor applied to the window.

```
- (CGFloat)userSpaceScaleFactor
```

Return Value

The scale factor applied to the window.

Discussion

Clients can multiply view coordinates by the returned scale factor to get a set of new coordinates that are scaled to the resolution of the target screen. For example, if the scale factor is 1.25 and the view frame size is 80 x 80, the actual size of the view frame is 100 x 100 pixels on the target screen.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

VideoViewer

Declared In

NSWindow.h

validRequestorForSendType:returnType:

Searches for an object that responds to a Services request.

```
- (id)validRequestorForSendType:(NSString *)sendType returnType:(NSString
    *)returnType
```

Parameters

sendType

The input type of the Services request.

returnType

The return type of the Services request.

Return Value

The object that responds to the services request; `nil` when none is found.

Discussion

Messages to perform this method are initiated by the Services menu. It's part of the mechanism that passes `validRequestorForSendType:returnType:` messages up the responder chain.

This method works by forwarding the message to the window's delegate if it responds (and provided it isn't an `NSResponder` object with its own next responder). If the delegate doesn't respond to the message or returns `nil` when sent it, this method forwards the message to the `NSApplication` object. If the `NSApplication` object returns `nil`, this method also returns `nil`. Otherwise this method returns the object returned by the delegate or the `NSApplication` object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `validRequestorForSendType:returnType:` (page 2204) (`NSResponder`)
- `validRequestorForSendType:returnType:` (page 180) (`NSApplication`)

Declared In

`NSWindow.h`

viewsNeedDisplay

Indicates whether any of the window's views need to be displayed.

```
- (BOOL)viewsNeedDisplay
```

Return Value

`YES` when any of the window's views need to be displayed; otherwise, `NO`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setViewsNeedDisplay:` (page 3400)

Declared In

`NSWindow.h`

windowController

Returns the window's window controller.

```
- (id)windowController
```

Return Value
The window's window controller.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setWindowController:` (page 3400)

Related Sample Code
FunHouse
PDFKitLinker2
QTAudioContextInsert
Sketch-112

Declared In
`NSWindow.h`

windowNumber

Provides the window number of the window's window device.

```
- (NSInteger)windowNumber
```

Return Value
The window number of the window's window device.

Discussion
Each window device in an application is given a unique window number—note that this isn't the same as the global window number assigned by the window server. This number can be used to identify the window device with the `orderWindow:relativeTo:` (page 3352) method and in the Application Kit function `NSWindowList` (page 4003). .

If the window doesn't have a window device, the value returned will be equal to or less than 0.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- initWithContentRect:styleMask:backing:defer:` (page 3332)
`- setOneShot:` (page 3391)

Related Sample Code
CocoaDVDPlayer
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSWindow.h`

windowRef

Returns the Carbon `WindowRef` associated with the window, creating one if necessary.

```
- (void *)windowRef
```

Discussion
This method can be used to create a `WindowRef` for a window containing a Carbon control. Subsequent calls to this method return the existing `WindowRef`. You use a `WindowRef` to create a Carbon window reference for a Cocoa window; this assists the integration of Carbon and Cocoa code and objects.

For more information see `MacWindows.h`. For more information on Carbon-Cocoa integration, see *Carbon-Cocoa Integration Guide*.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- initWithWindowRef:` (page 3334)

Declared In
`NSWindow.h`

worksWhenModal

Indicates whether the window is able to receive keyboard and mouse events even when some other window is being run modally.

```
- (BOOL)worksWhenModal
```

Return Value
`YES` if the window is able to receive keyboard and mouse events even when some other window is being run modally; otherwise, `NO`.

Discussion
The `NSWindow` implementation of this method returns `NO`. Only subclasses of `NSPanel` should override this default.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setWorksWhenModal:` (page 1862) (`NSPanel`)

Declared In
`NSWindow.h`

zoom:

This action method toggles the size and location of the window between its standard state (provided by the application as the "best" size to display the window's data) and its user state (a new size and location the user may have set by moving or resizing the window).

```
- (void)zoom:(id)sender
```

Parameters

sender

The object sending the message.

Discussion

For more information on the standard and user states, see `windowWillUseStandardFrame:defaultFrame:` (page 3942).

The `zoom:` method is typically invoked after a user clicks the window's zoom box but may also be invoked programmatically from the `performZoom:` (page 3355) method. It performs the following steps:

1. Invokes the `windowWillUseStandardFrame:defaultFrame:` (page 3942) method, if the delegate or the window class implements it, to obtain a "best fit" frame for the window. If neither the delegate nor the window class implements the method, uses a default frame that nearly fills the current screen, which is defined to be the screen containing the largest part of the window's current frame.
2. Adjusts the resulting frame, if necessary, to fit on the current screen.
3. Compares the resulting frame to the current frame to determine whether the window's standard frame is currently displayed. If the current frame is within a few pixels of the standard frame in size and location, it is considered a match.
4. Determines a new frame. If the window is currently in the standard state, the new frame represents the user state, saved during a previous zoom. If the window is currently in the user state, the new frame represents the standard state, computed in step 1 above. If there is no saved user state because there has been no previous zoom, the size and location of the window do not change.
5. Determines whether the window currently allows zooming. By default, zooming is allowed. If the window's delegate implements the `windowShouldZoom:toFrame:` (page 3938) method, `zoom:` invokes that method. If the delegate doesn't implement the method but the window does, `zoom:` invokes the window's version. `windowShouldZoom:toFrame:` returns `NO` if zooming is not currently allowed.
6. If the window currently allows zooming, sets the new frame.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- isZoomed` (page 3342)

Related Sample Code

CIAnnotation

Declared In

`NSWindow.h`

Constants

Window Style Masks

These constants specify the presence of a title and various buttons in a window's border. It can be `NSBorderlessWindowMask`, or it can contain any of the following options, combined using the C bitwise OR operator:

```
enum {
   NSBorderlessWindowMask = 0,
   NSTitledWindowMask = 1 << 0,
   NSClosableWindowMask = 1 << 1,
   NSMiniaturizableWindowMask = 1 << 2,
   NSResizableWindowMask = 1 << 3,
   NSTexturedBackgroundWindowMask = 1 << 8
};
```

Constants

`NSBorderlessWindowMask`

The window displays none of the usual peripheral elements. Useful only for display or caching purposes.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSTitledWindowMask`

The window displays a title bar.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSClosableWindowMask`

The window displays a close button.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSMiniaturizableWindowMask`

The window displays a minimize button.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSResizableWindowMask`

The window displays a resize control.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSTexturedBackgroundWindowMask`

The window displays with a metal-textured background. Additionally, the window may be moved by clicking and dragging anywhere in the window background. A bordered window with this mask gets rounded bottom corners.

Available in Mac OS X v10.2 and later.

Declared in `NSWindow.h`.

Declared In

`NSWindow.h`

Window Levels

The standard window levels in Mac OS X.

```
#define NSNormalWindowLevel           kCGNormalWindowLevel
#define NSFloatingWindowLevel         kCGFloatingWindowLevel
#define NSSubmenuWindowLevel          kCGTornOffMenuWindowLevel
#define NSTornOffMenuWindowLevel      kCGTornOffMenuWindowLevel
#define NSMainMenuWindowLevel         kCGMainMenuWindowLevel
#define NSStatusWindowLevel           kCGStatusWindowLevel
#define NSModalPanelWindowLevel       kCGModalPanelWindowLevel
#define NSPopUpMenuWindowLevel        kCGPopUpMenuWindowLevel
#define NSScreenSaverWindowLevel      kCGScreenSaverWindowLevel
#define NSDockWindowLevel             kCGDockWindowLevel
```

Constants

`NSNormalWindowLevel`

The default level for `NSWindow` objects.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSFloatingWindowLevel`

Useful for floating palettes.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSSubmenuWindowLevel`

Reserved for submenus. Synonymous with `NSTornOffMenuWindowLevel`, which is preferred.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSTornOffMenuWindowLevel`

The level for a torn-off menu. Synonymous with `NSSubmenuWindowLevel`.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSModalPanelWindowLevel`

The level for a modal panel.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSMainMenuWindowLevel`

Reserved for the application's main menu.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSStatusWindowLevel`

The level for a status window.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSPopUpMenuWindowLevel`
The level for a pop-up menu.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSScreenSaverWindowLevel`
The level for a screen saver.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSDockWindowLevel`

The level for the doc.. (Deprecated. Deprecated. There is no replacement.)

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

Discussion
The stacking of levels takes precedence over the stacking of windows within each level. That is, even the bottom window in a level will obscure the top window of the next level down. Levels are listed in order from lowest to highest. These constants are mapped (using `#define` statements) to corresponding elements in `Window Level Keys`.

Display Device—Descriptions

These constants are the keys for device description dictionaries used by `deviceDescription` (page 3318).

```
NSString *NSDeviceResolution;
NSString *NSDeviceColorSpaceName;
NSString *NSDeviceBitsPerSample;
NSString *NSDeviceIsScreen;
NSString *NSDeviceIsPrinter;
NSString *NSDeviceSize;
```

Constants
`NSDeviceResolution`
The corresponding value is an `NSValue` object containing a value of type `NSSize` that describes the window's raster resolution in dots per inch (dpi).

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSDeviceColorSpaceName`
The corresponding value is an `NSString` object giving the name of the window's color space.

See `Color_Space_Names` (page 4017) in *Application Kit Constants Reference* for a list of possible values.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSDeviceBitsPerSample`
The corresponding value is an `NSNumber` object containing an integer that gives the bit depth of the window's raster image (2-bit, 8-bit, and so forth).

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSDeviceIsScreen`
If there is a corresponding value, this indicates that the display device is a screen.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSDeviceIsPrinter`
If there is a corresponding value, this indicates that the display device is a printer.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSDeviceSize`
The corresponding value is an `NSValue` object containing a value of type `NSSize` that gives the size of the window's frame rectangle.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

Declared In
`NSGraphics.h`

Managing Scaling Factors

This constant provides a way to manage scaling factors:

```
enum {
    NSUnscaledWindowMask        = 1 << 11
};
```

Constants
`NSUnscaledWindowMask`
Specifies that the window is created without any scaling factors applied.

The client is responsible for all scaling operations in the window. Such a window returns `1.0` from its `userSpaceScaleFactor` method.

Currently restricted to borderless windows (`NSBorderlessWindowMask`).

Available in Mac OS X v10.4 and later.

Declared in `NSWindow.h`.

Controlling the Look of a Window and Its Toolbar

This constant controls the look of a window and its toolbar.

```
enum {
    NSUnifiedTitleAndToolbarWindowMask    = 1 << 12
};
```

Constants
`NSUnifiedTitleAndToolbarWindowMask`
Specifies a window whose toolbar and title bar are rendered on a single continuous background.

Available in Mac OS X v10.4 and later.

Declared in `NSWindow.h`.

NSSelectionDirection—Direction of Key View Change

These constants specify the direction a window is currently using to change the key view. They're used by `keyViewSelectionDirection` (page 3343).

```
enum {
    NSDirectSelection = 0,
    NSSelectingNext,
    NSSelectingPrevious
};
typedef NSUInteger NSSelectionDirection;
```

Constants

`NSDirectSelection`
The window isn't traversing the key view loop.
Available in Mac OS X v10.0 and later.
Declared in `NSWindow.h`.

`NSSelectingNext`
The window is proceeding to the next valid key view.
Available in Mac OS X v10.0 and later.
Declared in `NSWindow.h`.

`NSSelectingPrevious`
The window is proceeding to the previous valid key view.
Available in Mac OS X v10.0 and later.
Declared in `NSWindow.h`.

NSWindowButton—Accessing Standard Title Bar Buttons

These constants provide a way to access standard title bar buttons:

```
enum {
     NSWindowCloseButton,
     NSWindowMiniaturizeButton,
     NSWindowZoomButton,
     NSWindowToolbarButton,
     NSWindowDocumentIconButton
};
typedef NSUInteger NSWindowButton;
```

Constants

`NSWindowCloseButton`
The close button.
Available in Mac OS X v10.2 and later.
Declared in `NSWindow.h`.

`NSWindowMiniaturizeButton`
The minimize button.
Available in Mac OS X v10.2 and later.
Declared in `NSWindow.h`.

`NSWindowZoomButton`
The zoom button.

Available in Mac OS X v10.2 and later.

Declared in `NSWindow.h`.

`NSWindowToolbarButton`
The toolbar button.

Available in Mac OS X v10.2 and later.

Declared in `NSWindow.h`.

`NSWindowDocumentIconButton`
The document icon button.

Available in Mac OS X v10.2 and later.

Declared in `NSWindow.h`.

NSRunLoop—Ordering Modes for NSWindow

These constants are passed to `NSRunLoop`'s `performSelector:target:argument:order:modes:`.

```
enum {
   NSDisplayWindowRunLoopOrdering,
   NSResetCursorRectsRunLoopOrdering
};
```

Constants

`NSDisplayWindowRunLoopOrdering`
The priority at which windows are displayed.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

`NSResetCursorRectsRunLoopOrdering`
The priority at which cursor rects are reset.

Available in Mac OS X v10.0 and later.

Declared in `NSWindow.h`.

NSWindowDepth—Window Depth

This type represents the depth, or amount of memory, devoted to a single pixel in a window or screen. A depth of 0 indicates default depth. Window depths should not be made persistent as they will not be the same across systems.

```
typedef int NSWindowDepth;
```

Discussion

Use the functions `NSColorSpaceFromDepth` (page 3966), `NSBitsPerPixelFromDepth` (page 3966), and `NSPlanarFromDepth` (page 3990) to extract info from an `NSWindowDepth` value. Use `NSBestDepth` (page 3965) to compute window depths. `NSBestDepth` (page 3965) tries to accommodate all the parameters (match or better); if there are multiple matches, it gives the closest, with matching color space first, then bps, then planar, then `bpp`. `bpp` is "bits per pixel"; 0 indicates default (same as the number of bits per plane, either `bps` or `bps` * `NSNumberOfColorComponents` (page 3988)); other values maybe used as hints to provide backing stores of different configuration: for instance, 8-bit color.

On Mac OS X 10.6 and later, you can pass one of the explicit bit depths defined in "Explicit Window Depth Limits" (page 3417) to the NSWindow method `setDepthLimit:` (page 3378).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphics.h`

Explicit Window Depth Limits

These constants define explicit window depths that can be used with `setDepthLimit:` (page 3378).

```
enum {
    NSWindowDepthTwentyfourBitRGB = 0x208,
    NSWindowDepthSixtyfourBitRGB = 0x210,
    NSWindowDepthOnehundredtwentyeightBitRGB = 0x220
};
```

Constants

`NSWindowDepthTwentyfourBitRGB`
Twenty four bit RGB depth limit.
Available in Mac OS X v10.6 and later.
Declared in `NSGraphics.h`.

`NSWindowDepthSixtyfourBitRGB`
Sixty four bit RGB depth limit.
Available in Mac OS X v10.6 and later.
Declared in `NSGraphics.h`.

`NSWindowDepthOnehundredtwentyeightBitRGB`
One hundred and twenty eight bit RGB depth limit.
Available in Mac OS X v10.6 and later.
Declared in `NSGraphics.h`.

NSBackingStoreType—Buffered Window Drawing

These constants specify how the drawing done in a window is buffered by the window device.

```
enum {
    NSBackingStoreRetained     = 0,
    NSBackingStoreNonretained  = 1,
    NSBackingStoreBuffered     = 2
};
typedef NSUInteger NSBackingStoreType;
```

Constants

`NSBackingStoreRetained`

The window uses a buffer, but draws directly to the screen where possible and to the buffer for obscured portions.

You should not use this mode. It combines the limitations of `NSBackingStoreNonretained` with the memory use of `NSBackingStoreBuffered`. The original NeXTSTEP implementation was an interesting compromise that worked well with fast memory mapped framebuffers on the CPU bus—something that hasn't been in general use since around 1994. These tend to have performance problems.

In Mac OS X 10.5 and later, requests for retained windows will result in the window system creating a buffered window, as that better matches actual use.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSBackingStoreNonretained`

The window draws directly to the screen without using any buffer.

You should not use this mode. It exists primarily for use in the original Classic Blue Box. It does not support Quartz drawing, alpha blending, or opacity. Moreover, it does not support hardware acceleration, and interferes with system-wide display acceleration. If you use this mode, your application must manage visibility region clipping itself, and manage repainting on visibility changes.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSBackingStoreBuffered`

The window renders all drawing into a display buffer and then flushes it to the screen.

You should use this mode. It supports hardware acceleration, Quartz drawing, and takes advantage of the GPU when possible. It also supports alpha channel drawing, opacity controls, using the compositor.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSWindowOrderingMode

These constants let you specify how a window is ordered relative to another window. For more information, see `orderWindow:relativeTo:` (page 3352).

```
enum {
    NSWindowAbove          =  1,
    NSWindowBelow          = -1,
    NSWindowOut              =  0
};
typedef NSInteger NSWindowOrderingMode;
```

Constants

`NSWindowAbove`

Moves the window above the indicated window.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSWindowBelow`

Moves the window below the indicated window.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSWindowOut`

Moves the window off the screen.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSWindowSharingType

The following constants and the related data type represent the access levels other processes can have to a window's content.

```
enum {
    NSWindowSharingNone = 0,
    NSWindowSharingReadOnly = 1,
    NSWindowSharingReadWrite = 2
};
typedef NSUInteger NSWindowSharingType;
```

Constants

`NSWindowSharingNone`

The window's contents cannot be read by another process.

Available in Mac OS X v10.5 and later.

Declared in `NSWindow.h`.

`NSWindowSharingReadOnly`

The window's contents can be read but not modified by another process.

Available in Mac OS X v10.5 and later.

Declared in `NSWindow.h`.

`NSWindowSharingReadWrite`

The window's contents can be read and modified by another process.

Available in Mac OS X v10.5 and later.

Declared in `NSWindow.h`.

NSWindowBackingLocation

The following constants and the related data type represent a window's possible backing locations.

```
enum {
    NSWindowBackingLocationDefault = 0,
    NSWindowBackingLocationVideoMemory = 1,
    NSWindowBackingLocationMainMemory = 2
};
typedef NSUInteger NSWindowBackingLocation;
```

Constants

`NSWindowBackingLocationDefault`

Determined by the operating system.

Available in Mac OS X v10.5 and later.

Declared in `NSWindow.h`.

`NSWindowBackingLocationVideoMemory`

Video memory.

Available in Mac OS X v10.5 and later.

Declared in `NSWindow.h`.

`NSWindowBackingLocationMainMemory`

Physical memory.

Available in Mac OS X v10.5 and later.

Declared in `NSWindow.h`.

NSWindowNumberListOptions

The options that may be passed to the `windowNumbersWithOptions:` (page 3296) method.

```
enum {
   NSWindowNumberListAllApplications = 1 << 0,
   NSWindowNumberListAllSpaces = 1 << 4
};
typedef NSUInteger NSWindowNumberListOptions;
```

Constants

`NSWindowNumberListAllApplications`

The window numbers of windows visible on any space and belonging to any application.

Available in Mac OS X v10.6 and later.

Declared in `NSWindow.h`.

`NSWindowNumberListAllSpaces`

The window numbers of windows visible on any space and belonging to the calling application.

Available in Mac OS X v10.6 and later.

Declared in `NSWindow.h`.

Discussion

If the value 0 is passed instead, then the list returned from the method contains window numbers for visible windows on the active space belonging to the calling application.

Managing Window Collections

Window collection behaviors related to Exposé and Spaces.

```
enum {
   NSWindowCollectionBehaviorDefault = 0,
   NSWindowCollectionBehaviorCanJoinAllSpaces = 1 << 0,
   NSWindowCollectionBehaviorMoveToActiveSpace = 1 << 1
};
enum {
   NSWindowCollectionBehaviorManaged = 1 << 2,
   NSWindowCollectionBehaviorTransient = 1 << 3,
   NSWindowCollectionBehaviorStationary = 1 << 4,
};
enum {
   NSWindowCollectionBehaviorParticipatesInCycle = 1 << 5,
   NSWindowCollectionBehaviorIgnoresCycle = 1 << 6
};
typedef NSUInteger NSWindowCollectionBehavior;
```

Constants

`NSWindowCollectionBehaviorDefault`

The window can be associated to one space at a time.

Available in Mac OS X v10.5 and later.

Declared in `NSWindow.h`.

`NSWindowCollectionBehaviorCanJoinAllSpaces`

The window appears in all spaces. The menu bar behaves this way.

Available in Mac OS X v10.5 and later.

Declared in `NSWindow.h`.

`NSWindowCollectionBehaviorMoveToActiveSpace`

Making the window active does not cause a space switch; the window switches to the active space.

Available in Mac OS X v10.5 and later.

Declared in `NSWindow.h`.

`NSWindowCollectionBehaviorManaged`

The window participates in Spaces and Exposé. This is the default behavior if `windowLevel` is equal to `NSNormalWindowLevel` (page 3412).

Available in Mac OS X v10.6 and later.

Declared in `NSWindow.h`.

`NSWindowCollectionBehaviorTransient`

The window floats in Spaces and is hidden by Exposé. This is the default behavior if `windowLevel` is not equal to `NSNormalWindowLevel` (page 3412).

Available in Mac OS X v10.6 and later.

Declared in `NSWindow.h`.

`NSWindowCollectionBehaviorStationary`

The window is unaffected by Exposé; it stays visible and stationary, like the desktop window.

Available in Mac OS X v10.6 and later.

Declared in `NSWindow.h`.

`NSWindowCollectionBehaviorParticipatesInCycle`
The window participates in the window cycle for use with the Cycle Through Windows Window menu item.

Available in Mac OS X v10.6 and later.

Declared in `NSWindow.h`.

`NSWindowCollectionBehaviorIgnoresCycle`
The window is not part of the window cycle for use with the Cycle Through Windows Window menu item.

Available in Mac OS X v10.6 and later.

Declared in `NSWindow.h`.

Application Kit Version for Deferred Window Display Support

The version of the AppKit.framework containing a specific bug fix or capability.

```
#define NSAppKitVersionNumberWithDeferredWindowDisplaySupport        1019.0
```

Constants

`NSAppKitVersionNumberWithDeferredWindowDisplaySupport`
The specific version of the AppKit framework that introduced for custom sheet positioning. Developers should not need to use this constant unless they are writing applications for Mac OS X v10.5 and earlier.

Available in Mac OS X v10.6 and later.

Declared in `NSWindow.h`.

Application Kit Version for Custom Sheet Position

The version of the AppKit.framework containing a specific bug fix or capability.

```
#define NSAppKitVersionNumberWithCustomSheetPosition 686.0
```

Constants

`NSAppKitVersionNumberWithCustomSheetPosition`
The specific version of the AppKit framework that introduced for custom sheet positioning. Developers should not need to use this constant unless they are writing applications for Mac OS X v10.2 and earlier.

Available in Mac OS X v10.3 and later.

Declared in `NSWindow.h`.

Notifications

NSWindowDidBecomeKeyNotification

Posted whenever an `NSWindow` object becomes the key window.

The notification object is the `NSWindow` object that has become key. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

NSWindowDidBecomeMainNotification

Posted whenever an `NSWindow` object becomes the main window.

The notification object is the `NSWindow` object that has become main. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

NSWindowDidChangeScreenNotification

Posted whenever a portion of an `NSWindow` object's frame moves onto or off of a screen.

The notification object is the `NSWindow` object that has changed screens. This notification does not contain a *`userInfo`* dictionary.

This notification is not sent in Mac OS X versions earlier than 10.4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

NSWindowDidChangeScreenProfileNotification

Posted whenever the display profile for the screen containing the window changes.

This notification is sent only if the window returns `YES` from `displaysWhenScreenProfileChanges` (page 3322). This notification may be sent when a majority of the window is moved to a different screen (whose profile is also different from the previous screen) or when the ColorSync profile for the current screen changes.

The notification object is the `NSWindow` object whose profile changed. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSWindow.h`

NSWindowDidDeminiaturizeNotification

Posted whenever an `NSWindow` object is deminimized.

The notification object is the `NSWindow` object that has been deminimized. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

NSWindowDidEndSheetNotification

Posted whenever an `NSWindow` object closes an attached sheet.

The notification object is the `NSWindow` object that contained the sheet. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.1 and later.

Declared In
`NSWindow.h`

NSWindowDidEndLiveResizeNotification

Posted after the user resizes a window.

This notification is sent only once for a series of window resize operations.

The notification object is the `NSWindow` object that was resized. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSWindow.h`

NSWindowDidExposeNotification

Posted whenever a portion of a nonretained `NSWindow` object is exposed, whether by being ordered in front of other windows or by other windows being removed from in front of it.

The notification object is the `NSWindow` object that has been exposed. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSExposedRect"`	The rectangle that has been exposed (an `NSValue` object containing an NSRect).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

NSWindowDidMiniaturizeNotification

Posted whenever an NSWindow object is minimized.

The notification object is the NSWindow object that has been minimized. This notification does not contain a *userInfo* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

NSWindowDidMoveNotification

Posted whenever an NSWindow object is moved.

The notification object is the NSWindow object that has moved. This notification does not contain a *userInfo* dictionary.

Note: This notification is sent when the window that moved didn't also change size. See NSWindowDidResizeNotification (page 3426) for more information.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

NSWindowDidResignKeyNotification

Posted whenever an NSWindow object resigns its status as key window.

The notification object is the NSWindow object that has resigned its key window status. This notification does not contain a *userInfo* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

NSWindowDidResignMainNotification

Posted whenever an NSWindow object resigns its status as main window.

The notification object is the NSWindow object that has resigned its main window status. This notification does not contain a *userInfo* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

NSWindowDidResizeNotification

Posted whenever an `NSWindow` object's size changes.

The notification object is the `NSWindow` object whose size has changed. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

NSWindowDidUpdateNotification

Posted whenever an `NSWindow` object receives an `update` (page 3405) message.

The notification object is the `NSWindow` object that received the `update` (page 3405) message. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindow.h`

NSWindowWillBeginSheetNotification

Posted whenever an `NSWindow` object is about to open a sheet.

The notification object is the `NSWindow` object that is about to open the sheet. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.1 and later.

Declared In
`NSWindow.h`

NSWindowWillCloseNotification

Posted whenever an `NSWindow` object is about to close.

The notification object is the `NSWindow` object that is about to close. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

NSWindowWillMiniaturizeNotification

Posted whenever an NSWindow object is about to be minimized.

The notification object is the NSWindow object that is about to be minimized. This notification does not contain a *userInfo* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

NSWindowWillMoveNotification

Posted whenever an NSWindow object is about to move.

The notification object is the NSWindow object that is about to move. This notification does not contain a *userInfo* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindow.h

NSWindowWillStartLiveResizeNotification

Posted before the user resizes a window.

This notification is sent only once for a series of window resize operations.

The notification object is the NSWindow object that is about to be live resized. This notification does not contain a *userInfo* dictionary.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSWindow.h

NSWindowController Class Reference

Inherits from	NSResponder : NSObject
Conforms to	NSCoding NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSWindowController.h
Companion guide	Document-Based Applications Overview
Related sample code	BundleLoader ImageKitDemo QTAudioContextInsert Sketch+Accessibility Sketch-112

Overview

An `NSWindowController` object manages a window, usually a window stored in a nib file.

This management entails:

- Loading and displaying the window
- Closing the window when appropriate
- Customizing the window's title
- Storing the window's frame (size and location) in the defaults database
- Cascading the window in relation to other document windows of the application

A window controller can manage a window by itself or as a role player in the Application Kit's document-based architecture, which also includes `NSDocument` and `NSDocumentController` objects. In this architecture, a window controller is created and managed by a "document" (an instance of an `NSDocument` subclass) and, in turn, keeps a reference to the document.

The relationship between a window controller and a nib file is important. Although a window controller can manage a programmatically created window, it usually manages a window in a nib file. The nib file can contain other top-level objects, including other windows, but the window controller's responsibility is this

primary window. The window controller is usually the owner of the nib file, even when it is part of a document-based application. Regardless of who is the file's owner, the window controller is responsible for freeing all top-level objects in the nib file it loads.

For simple documents—that is, documents with only one nib file containing a window—you need do little directly with `NSWindowController`. The Application Kit will create one for you. However, if the default window controller is not sufficient, you can create a custom subclass of `NSWindowController`. For documents with multiple windows or panels, your document must create separate instances of `NSWindowController` (or of custom subclasses of `NSWindowController`), one for each window or panel. An example is a CAD application that has different windows for side, top, and front views of drawn objects. What you do in your `NSDocument` subclass determines whether the default `NSWindowController` or separately created and configured `NSWindowController` objects are used.

Subclassing NSWindowController

You should create a subclass of `NSWindowController` when you want to augment the default behavior, such as to give the window a custom title or to perform some setup tasks before the window is loaded. In your class's initialization method, be sure to invoke on `super` either one of the `initWithWindowNibName:...` initializers or the `initWithWindow:` (page 3434) initializer. Which one depends on whether the window object originates in a nib file or is programmatically created.

Three `NSWindowController` methods are most commonly overridden:

Method Name	Description
`windowWillLoad` (page 3444)	Override to perform tasks before the window nib file is loaded.
`windowDidLoad` (page 3442)	Override to perform tasks after the window nib file is loaded.
`windowTitleForDocumentDisplayName:` (page 3444)	Override to customize the window title.

You can also override `loadWindow` (page 3436) to get different nib-searching or nib-loading behavior, although there is usually no need to do this.

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Initializing NSWindowControllers

- `initWithWindow:` (page 3434)
  Returns a window controller initialized with a given window.
- `initWithWindowNibName:` (page 3434)
  Returns a window controller initialized with a nib file.
- `initWithWindowNibName:owner:` (page 3435)
  Returns a window controller initialized with a nib file and a specified owner for that nib file.
- `initWithWindowNibPath:owner:` (page 3435)
  Returns a window controller initialized with a nib file at an absolute path and a specified owner.

Loading and Display the Window

- `loadWindow` (page 3436)
  Loads the receiver's window from the nib file.
- `showWindow:` (page 3441)
  Displays the window associated with the receiver.
- `isWindowLoaded` (page 3436)
  Returns whether the nib file containing the receiver's window has been loaded.
- `window` (page 3441)
  Returns the window owned by the receiver.
- `setWindow:` (page 3439)
  Sets the window controller's window.
- `windowDidLoad` (page 3442)
  Sent after the window owned by the receiver has been loaded.
- `windowWillLoad` (page 3444)
  Sent before the window owned by the receiver is loaded.

Setting and Getting the Document

- `setDocument:` (page 3437)
  Sets the document associated with the window managed by the receiver.
- `document` (page 3433)
  Returns the document associated with the receiver.
- `setDocumentEdited:` (page 3437)
  Sets the document edited flag for the window controller.

Closing the Window

- `close` (page 3432)
  Closes the window if it was loaded.
- `shouldCloseDocument` (page 3440)
  Returns whether the receiver necessarily closes the associated document when the window it manages is closed.
- `setShouldCloseDocument:` (page 3438)
  Sets whether the receiver should necessarily close the associated document when the window it manages is closed.

Getting Nib File Information

- `owner` (page 3437)
  Returns the owner of the nib file containing the window managed by the receiver.
- `windowNibName` (page 3443)
  Returns the name of the nib file that stores the window associated with the receiver.
- `windowNibPath` (page 3444)
  Returns the full path of the nib file that stores the window associated with the receiver.

Setting and Getting Window Attributes

- `setShouldCascadeWindows:` (page 3438)
  Sets whether the window should cascade in relation to other document windows.
- `shouldCascadeWindows` (page 3440)
  Returns whether the window will cascade in relation to other document windows when it is displayed.
- `setWindowFrameAutosaveName:` (page 3439)
  Sets the name under which the window's frame is saved in the defaults database.
- `windowFrameAutosaveName` (page 3443)
  Returns the name under which the frame rectangle of the window owned by the receiver is stored in the defaults database.
- `synchronizeWindowTitleWithDocumentName` (page 3441)
  Synchronizes the displayed window title and the represented filename with the information in the associated document.
- `windowTitleForDocumentDisplayName:` (page 3444)
  Returns the window title to be used for a given document display name.

Instance Methods

close

Closes the window if it was loaded.

```
- (void)close
```

Discussion
Because this method closes the window without asking the user for confirmation, you usually do not invoke it when the Close menu command is chosen. Instead invoke NSWindow's `performClose:` (page 3354) on the receiver's window. See "Window Closing Behavior" for an overview of deallocation behavior when a window is closed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `shouldCloseDocument` (page 3440)
- `setShouldCloseDocument:` (page 3438)

Related Sample Code
FunHouse
GLUT
QTCompressionOptionsWindow
UIElementInspector

Declared In
`NSWindowController.h`

document

Returns the document associated with the receiver.

```
- (id)document
```

Return Value
The document associated with the receiver or `nil` if there is none.

Discussion
When a window controller is added to a document's list of window controllers, the document sets the window controller's document with `setDocument:`. The Application Kit uses this outlet to access the document for relevant next-responder messages.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setDocument:` (page 3437)

Related Sample Code
EnhancedAudioBurn
PDFKitLinker2
QTAudioContextInsert
QTKitFrameStepper
Sketch-112

Declared In
`NSWindowController.h`

initWithWindow:

Returns a window controller initialized with a given window.

```
- (id)initWithWindow:(NSWindow *)window
```

Parameters

window

The window object to manage; can be `nil`.

Return Value

A newly initialized window controller.

Discussion

This method is the designated initializer for `NSWindowController`.

This initializer is useful when a window has been loaded but no window controller is assigned. The default initialization turns on cascading, sets the `shouldCloseDocument` (page 3440) flag to `NO`, and sets the window frame autosave name to an empty string. As a side effect, the created window controller is added as an observer of the `NSWindowWillCloseNotification` (page 3426)s posted by that window object (which is handled by a private method). If you make the window controller a delegate of the window, you can implement NSWindow's `windowShouldClose:` delegate method.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

FunHouse

MyMediaPlayer

UIElementInspector

Declared In

`NSWindowController.h`

initWithWindowNibName:

Returns a window controller initialized with a nib file.

```
- (id)initWithWindowNibName:(NSString *)windowNibName
```

Parameters

windowNibName

The name of the nib file (minus the "`.nib`" extension) that archives the receiver's window; cannot be `nil`.

Discussion

Sets the owner of the nib file to the receiver. The default initialization turns on cascading, sets the `shouldCloseDocument` (page 3440) flag to `NO`, and sets the autosave name for the window's frame to an empty string.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

BundleLoader

DemoMonkey
QTAudioContextInsert
Sketch+Accessibility
UIElementInspector

Declared In
NSWindowController.h

initWithWindowNibName:owner:

Returns a window controller initialized with a nib file and a specified owner for that nib file.

```
- (id)initWithWindowNibName:(NSString *)windowNibName owner:(id)owner
```

Parameters

windowNibName
The name of the nib file (minus the ".nib" extension) that archives the receiver's window; cannot be nil.

owner
The nib file's owner; cannot be nil.

Discussion
The default initialization turns on cascading, sets the shouldCloseDocument (page 3440) flag to NO, and sets the autosave name for the window's frame to an empty string.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindowController.h

initWithWindowNibPath:owner:

Returns a window controller initialized with a nib file at an absolute path and a specified owner.

```
- (id)initWithWindowNibPath:(NSString *)windowNibPath owner:(id)owner
```

Parameters

windowNibPath
The full path to the nib file that archives the receiver's window; cannot be nil.

owner
The nib file's owner; cannot be nil.

Discussion
Use this method if your nib file is at a fixed location (which is not inside either the file's owner's class's bundle or in the application's main bundle). The default initialization turns on cascading, sets the shouldCloseDocument (page 3440) flag to NO, and sets the autosave name for the window's frame to an empty string.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindowController.h

isWindowLoaded

Returns whether the nib file containing the receiver's window has been loaded.

```
- (BOOL)isWindowLoaded
```

Return Value
YES if the nib file containing the receiver's window has been loaded, NO otherwise.

Availability
Available in Mac OS X v10.0 and later.

See Also
- loadWindow (page 3436)
- window (page 3441)
- windowDidLoad (page 3442)
- windowWillLoad (page 3444)

Related Sample Code
GLUT
QTCompressionOptionsWindow
Sketch-112

Declared In
NSWindowController.h

loadWindow

Loads the receiver's window from the nib file.

```
- (void)loadWindow
```

Discussion
You should never directly invoke this method. Instead, invoke window (page 3441) so the windowDidLoad (page 3442) and windowWillLoad (page 3444) methods are invoked. Subclasses can override this method if the way it finds and loads the window is not adequate. It uses NSBundle's bundleForClass: method to get the bundle, using the class of the nib file owner as argument. It then locates the nib file within the bundle and, if successful, loads it; if unsuccessful, it tries to find the nib file in the main bundle.

Availability
Available in Mac OS X v10.0 and later.

See Also
- isWindowLoaded (page 3436)

Declared In
NSWindowController.h

owner

Returns the owner of the nib file containing the window managed by the receiver.

```
- (id)owner
```

Return Value
The owner of the nib file containing the window managed by the receiver; usually `self`, but can be the receiver's document or some other object.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- windowNibName` (page 3443)

Related Sample Code
ImageKitDemo

Declared In
`NSWindowController.h`

setDocument:

Sets the document associated with the window managed by the receiver.

```
- (void)setDocument:(NSDocument *)document
```

Parameters
document
The new document.

Discussion
Documents automatically call this method when they add a window controller to their list of window controllers; you should not call it directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- document` (page 3433)

Declared In
`NSWindowController.h`

setDocumentEdited:

Sets the document edited flag for the window controller.

```
- (void)setDocumentEdited:(BOOL)flag
```

Parameters
flag
`YES` if the document has been edited since its last save, `NO` if it hasn't.

Discussion
The window controller uses this flag to control whether its associated window shows up as dirty. You should not call this method directly for window controllers with an associated document; the document calls this method on its window controllers as needed.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindowController.h

setShouldCascadeWindows:

Sets whether the window should cascade in relation to other document windows.

```
- (void)setShouldCascadeWindows:(BOOL)flag
```

Parameters

flag

YES if the window should cascade in relation to other document windows, NO otherwise.

Discussion
Cascading in relation to other document windows means having a slightly offset location so that the title bars of previously displayed windows are still visible.

The default is YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- shouldCascadeWindows (page 3440)

Related Sample Code
Sketch+Accessibility

Declared In
NSWindowController.h

setShouldCloseDocument:

Sets whether the receiver should necessarily close the associated document when the window it manages is closed.

```
- (void)setShouldCloseDocument:(BOOL)flag
```

Parameters

flag

YES if the receiver necessarily closes the associated document when the window it manages is closed, NO otherwise.

Discussion
If NO, the document is closed only when the last remaining window of the document is closed.

The default is NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- shouldCloseDocument (page 3440)

Related Sample Code
DemoMonkey

Declared In
NSWindowController.h

setWindow:

Sets the window controller's window.

```
- (void)setWindow:(NSWindow *)aWindow
```

Parameters

aWindow
The new window.

Discussion
This method releases the old window and any associated top-level objects in its nib file and assumes ownership of the new window. You should generally create a new window controller for a new window and release the old window controller instead of using this method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindowController.h

setWindowFrameAutosaveName:

Sets the name under which the window's frame is saved in the defaults database.

```
- (void)setWindowFrameAutosaveName:(NSString *)name
```

Parameters

name
The name under which the window's frame is saved in the defaults database.

Discussion
By default, *name* is an empty string, causing no information to be stored in the defaults database.

Availability
Available in Mac OS X v10.0 and later.

See Also
- windowFrameAutosaveName (page 3443)
- setFrameAutosaveName: (page 3382) (NSWindow)

Related Sample Code
Sketch+Accessibility

Declared In
NSWindowController.h

shouldCascadeWindows

Returns whether the window will cascade in relation to other document windows when it is displayed.

```
- (BOOL)shouldCascadeWindows
```

Return Value
YES if the window will cascade in relation to other document windows, NO otherwise.

Discussion
The default is YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setShouldCascadeWindows: (page 3438)

Declared In
NSWindowController.h

shouldCloseDocument

Returns whether the receiver necessarily closes the associated document when the window it manages is closed.

```
- (BOOL)shouldCloseDocument
```

Return Value
YES if the receiver necessarily closes the associated document when the window it manages is closed, NO otherwise.

Discussion
If NO, the document is closed only when the last remaining window of the document is closed.

The default is NO.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setShouldCloseDocument: (page 3438)

Declared In
NSWindowController.h

showWindow:

Displays the window associated with the receiver.

```
- (IBAction)showWindow:(id)sender
```

Parameters

sender

The control sending the message; can be nil.

Discussion

If the window is an `NSPanel` object and has its `becomesKeyOnlyIfNeeded` (page 1860) flag set to `YES`, the window is displayed in front of all other windows but is not made key; otherwise it is displayed in front and is made key. This method is useful for menu actions.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- makeKeyAndOrderFront:` (page 3345) (NSWindow)

`- orderFront:` (page 3351) (NSWindow)

Related Sample Code

BasicCocoaAnimations

FunHouse

QTAudioContextInsert

Sketch+Accessibility

Sketch-112

Declared In

`NSWindowController.h`

synchronizeWindowTitleWithDocumentName

Synchronizes the displayed window title and the represented filename with the information in the associated document.

```
- (void)synchronizeWindowTitleWithDocumentName
```

Discussion

Does nothing if the window controller has no associated document or loaded window. This method queries the window controller's document to get the document's display name and full filename path, then calls `windowTitleForDocumentDisplayName:` (page 3444) to get the display name to show in the window title.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSWindowController.h`

window

Returns the window owned by the receiver.

```
- (NSWindow *)window
```

Return Value
The window owned by the receiver or `nil` if there isn't one.

Discussion
If the window has not yet been loaded, this method attempts to load the window's nib file using `loadWindow` (page 3436). Before it loads the window, it invokes `windowWillLoad` (page 3444), and if the window controller has a document, it invokes the document's corresponding method `windowControllerWillLoadNib:` (page 991) (if implemented). After loading the window, this method invokes `windowDidLoad` (page 3442) and, if there is a document, the NSDocument method `windowControllerDidLoadNib:` (page 990) (if implemented).

Availability
Available in Mac OS X v10.0 and later.

See Also
- `windowControllerWillLoadNib:` (page 991) (NSDocument)

Related Sample Code
FunHouse
GridCalendar
ObjectPath
QTAudioContextInsert
UIElementInspector

Declared In
`NSWindowController.h`

windowDidLoad

Sent after the window owned by the receiver has been loaded.

```
- (void)windowDidLoad
```

Discussion
The default implementation does nothing.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `loadWindow` (page 3436)
- `window` (page 3441)
- `windowWillLoad` (page 3444)

Related Sample Code
FunHouse
GLUT
GridCalendar
QTAudioContextInsert
Sketch-112

Declared In
NSWindowController.h

windowFrameAutosaveName

Returns the name under which the frame rectangle of the window owned by the receiver is stored in the defaults database.

```
- (NSString *)windowFrameAutosaveName
```

Return Value
The name under which the frame rectangle of the window owned by the receiver is stored in the defaults database.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setWindowFrameAutosaveName: (page 3439)

Declared In
NSWindowController.h

windowNibName

Returns the name of the nib file that stores the window associated with the receiver.

```
- (NSString *)windowNibName
```

Return Value
The name of the nib file that stores the window associated with the receiver.

Discussion
If initWithWindowNibPath:owner: (page 3435) was used to initialize the instance, windowNibName returns the last path component with the ".nib" extension stripped off. If initWithWindowNibName: (page 3434) or initWithWindowNibName:owner: (page 3435) was used, windowNibName returns the name without the ".nib" extension.

Availability
Available in Mac OS X v10.0 and later.

See Also
- owner (page 3437)

Related Sample Code
ObjectPath
QTCompressionOptionsWindow
SourceView

Declared In
NSWindowController.h

windowNibPath

Returns the full path of the nib file that stores the window associated with the receiver.

```
- (NSString *)windowNibPath
```

Return Value
The full path of the nib file that stores the window associated with the receiver; `nil` if it cannot be located.

Discussion
If `initWithWindowNibPath:owner:` (page 3435) was used to initialize the instance, the path is just returned. If `initWithWindowNibName:` (page 3434) or `initWithWindowNibName:owner:` (page 3435) was used, `windowNibPath` locates the nib in the file's owner's class' bundle or in the application's main bundle and returns the full path (or `nil` if it cannot be located). Subclasses can override this to augment the search behavior, but probably ought to call `super` first.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindowController.h`

windowTitleForDocumentDisplayName:

Returns the window title to be used for a given document display name.

```
- (NSString *)windowTitleForDocumentDisplayName:(NSString *)displayName
```

Parameters

displayName
The display name for the document. This is the last path component under which the document file is saved.

Discussion
The default implementation returns *displayName*. Subclasses can override this method to customize the window title. For example, a CAD application could append "-Top" or "-Side," depending on the view displayed by the window.

Availability
Available in Mac OS X v10.0 and later.

See Also
`synchronizeWindowTitleWithDocumentName` (page 3441)

Declared In
`NSWindowController.h`

windowWillLoad

Sent before the window owned by the receiver is loaded.

```
- (void)windowWillLoad
```

Discussion
The default implementation does nothing.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `loadWindow` (page 3436)
- `window` (page 3441)
- `windowDidLoad` (page 3442)

Declared In
`NSWindowController.h`

NSWorkspace Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSWorkspace.h AppKit/NSRunningApplication.h
Companion guide	Workspace Services Programming Topics
Related sample code	AppList DesktopImage iChatStatusFromApplication Quartz Composer WWDC 2005 TextEdit SourceView

Overview

An `NSWorkspace` object responds to application requests to perform a variety of services:

- Opening, manipulating, and obtaining information about files and devices
- Tracking changes to the file system, devices, and the user database
- Getting and setting Finder information for files.
- Launching applications

There is one shared `NSWorkspace` object per application. You use the class method `sharedWorkspace` (page 3452) to access it. For example, the following statement uses an `NSWorkspace` object to request that a file be opened in the TextEdit application:

```
[[NSWorkspace sharedWorkspace] openFile:@"/Myfiles/README"
    withApplication:@"TextEdit"];
```

Tasks

Accessing the Shared NSWorkspace Instance

`+ sharedWorkspace` (page 3452)
Returns the shared `NSWorkspace` instance.

Accessing the NSWorkspace Notification Center

`- notificationCenter` (page 3469)
Returns the notification center for workspace notifications.

Opening Files

`- openFile:` (page 3469)
Opens the specified file specified using the default application associated with its type.

`- openFile:withApplication:` (page 3471)
Opens a file using the specified application.

`- openFile:fromImage:at:inView:` (page 3470)
Opens a file using the default application for its type and animates the action using a custom icon.

`- openFile:withApplication:andDeactivate:` (page 3471)
Opens the specified file and optionally deactivates the sending application.

`- openURL:` (page 3473)
Opens the location at the specified URL.

`- openTempFile:` (page 3472) Deprecated in Mac OS X v10.6
Opens the specified temporary file using the default application for its type.

Manipulating Applications

`- launchApplication:` (page 3463)
Launches the specified application.

`- launchApplication:showIcon:autolaunch:` (page 3463)
Launches the specified application using additional options.

`- launchApplicationAtURL:options:configuration:error:` (page 3464)
Launches the app at the specified URL.

`- hideOtherApplications` (page 3460)
Hides all applications other than the sender.

Manipulating Files

- `duplicateURLs:completionHandler:` (page 3455)
  Duplicates the specified URLS asynchronously in the same manner as the Finder..
- `recycleURLs:completionHandler:` (page 3475)
  Moves the specified URLs to the trash in the same manner a the Finder.
- `performFileOperation:source:destination:files:tag:` (page 3474)
  Performs a file operation on a set of files in a particular directory.
- `activateFileViewerSelectingURLs:` (page 3453)
  Activates the Finder, and opens one or more windows selecting the specified files.
- `selectFile:inFileViewerRootedAtPath:` (page 3477)
  Selects the file specified by *fullPath*.

Manipulating Uniform Type Identifier Information

- `typeOfFile:error:` (page 3480)
  Returns the uniform type identifier of the specified file, if it can be determined..
- `localizedDescriptionForType:` (page 3466)
  Returns the localized description for the specified Uniform Type Identifier
- `preferredFilenameExtensionForType:` (page 3475)
  Returns the preferred filename extension for the specified Uniform Type Identifier.
- `filenameExtension:isValidForType:` (page 3457)
  Returns whether the specified filename extension is appropriate for the Uniform Type Identifier.
- `type:conformsToType:` (page 3480)
  Returns a Boolean indicating that the first Uniform Type Identifier conforms to the second Uniform Type Identifier.
- `URLForApplicationWithBundleIdentifier:` (page 3482)
  Returns the URL for the application with the specified identifier.

Requesting Information

- `iconForFile:` (page 3460)
  Returns an image containing the icon for the specified file.
- `iconForFileType:` (page 3462)
  Returns an image containing the icon for files of the specified type.
- `iconForFiles:` (page 3461)
  Returns an image containing the icon for the specified files.
- `getInfoForFile:application:type:` (page 3460)
  Retrieves information about the specified file.
- `URLForApplicationToOpenURL:` (page 3482)
  Returns the URL to the default application that would be used to open the given URL.
- `fullPathForApplication:` (page 3458)
  Returns the full path for the specified application.

- `getFileSystemInfoForPath:isRemovable:isWritable:isUnmountable:description:type:` (page 3459)
  Describes the file system at *fullPath*.
- `isFilePackageAtPath:` (page 3462)
  Determines whether the specified path is a file package.
- `activeApplication` (page 3453)
  Returns a dictionary with information about the current active application.
- `launchedApplications` (page 3465)
  Returns an array of dictionaries, one entry for each running application.

Image Animation

- `slideImage:from:to:` (page 3479)
  Animates a sliding image from one point to another.

Requesting Additional Time Before Logout

- `extendPowerOffBy:` (page 3456)
  Requests the system wait for the specified amount of time before turning off the power or logging out the user.

Tracking Changes to the File System

- `noteFileSystemChanged:` (page 3468)
  Informs the `NSWorkspace` object that the file system changed at the specified path.
- `fileSystemChanged` (page 3458) Deprecated in Mac OS X v10.6
  Returns a Boolean value indicating whether a change to the file system has been registered with a `noteFileSystemChanged` (page 3468) message since the last `fileSystemChanged` (page 3458) message.
- `noteFileSystemChanged` (page 3468) Deprecated in Mac OS X v10.6
  Informs the `NSWorkspace` object that the file system has changed.

Updating Registered Services and File Types

- `findApplications` (page 3458) Deprecated in Mac OS X v10.6
  Examines all applications and updates the records of registered services and file types.

Tracking Changes to the Defaults Database

- `noteUserDefaultsChanged` (page 3468) Deprecated in Mac OS X v10.6
  Informs the `NSWorkspace` object that the defaults database has changed.

- `userDefaultsChanged` (page 3483) Deprecated in Mac OS X v10.6
  Returns a Boolean value indicating whether a change to the defaults database has been registered with a `noteUserDefaultsChanged` (page 3468) message since the last `userDefaultsChanged` (page 3483) message.

Tracking Status Changes for Applications and Devices

- `mountedRemovableMedia` (page 3466)
  Returns the full pathnames of all currently mounted removable disks.
- `mountedLocalVolumePaths` (page 3466)
  Returns the mount points of all local volumes, not just the removable ones returned by `mountedRemovableMedia` (page 3466).
- `runningApplications` (page 3476)
  Returns an array of `NSRunningApplication` representing the running applications.
- `checkForRemovableMedia` (page 3454) Deprecated in Mac OS X v10.6
  Polls the system's drives for any disks that have been inserted but not yet mounted.
- `mountNewRemovableMedia` (page 3467) Deprecated in Mac OS X v10.6
  Returns the full pathnames of any newly mounted disks.

Providing Custom Icons

- `setIcon:forFile:options:` (page 3478)
  Sets the icon for the file or directory at the specified path.

Unmounting a Device

- `unmountAndEjectDeviceAtPath:` (page 3481)
  Unmounts and ejects the device at the specified path.
- `unmountAndEjectDeviceAtURL:error:` (page 3481)
  Attempts to eject the volume mounted at the given path.

Working with Bundles

- `absolutePathForAppBundleWithIdentifier:` (page 3453)
  Returns the absolute file-system path of an application bundle.
- `launchAppWithBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifier:` (page 3464)
  Launches the application corresponding to the specified *`bundleIdentifier`*.
- `openURLs:withAppBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifiers:` (page 3473)
  Opens one or more files from an array of URLs.

Managing the Desktop Image

- `desktopImageURLForScreen:` (page 3455)
  Returns the URL for the desktop image for the given screen.
- `setDesktopImageURL:forScreen:options:error:` (page 3477)
  Sets the desktop image for the given screen to the image at the specified URL.
- `desktopImageOptionsForScreen:` (page 3454)
  Returns the desktop image options for the given screen.

Performing Finder Spotlight Searches

- `showSearchResultsForQueryString:` (page 3479)
  Displays a Spotlight search results window in Finder for the specified query string.

Finder File Labels

- `fileLabelColors` (page 3457)
  Returns the corresponding array of file label colors for the file labels.
- `fileLabels` (page 3457)
  Returns the array of file labels as strings.

Class Methods

sharedWorkspace

Returns the shared `NSWorkspace` instance.

```
+ (NSWorkspace *)sharedWorkspace
```

Return Value
The `NSWorkspace` object associated with the process.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaDVDPlayer
DesktopImage
iChatStatusFromApplication
Quartz Composer WWDC 2005 TextEdit
SourceView

Declared In
`NSWorkspace.h`

Instance Methods

absolutePathForAppBundleWithIdentifier:

Returns the absolute file-system path of an application bundle.

```
- (NSString *)absolutePathForAppBundleWithIdentifier:(NSString *)bundleIdentifier
```

Parameters

bundleIdentifier

The bundle identifier string. This value corresponds to the value in the `CFBundleIdentifier` key of the application's `Info.plist` file. For example, the bundle identifier of the TextEdit application is `com.apple.TextEdit`.

Return Value

The file system path to the application bundle identified by *bundleIdentifier*, or `nil` if the bundle cannot be found.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

`NSWorkspace.h`

activateFileViewerSelectingURLs:

Activates the Finder, and opens one or more windows selecting the specified files.

```
- (void)activateFileViewerSelectingURLs:(NSArray *)fileURLs
```

Parameters

fileURLs

The files to select and display in the Finder.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSWorkspace.h`

activeApplication

Returns a dictionary with information about the current active application.

```
- (NSDictionary *)activeApplication
```

Return Value

A dictionary with information about the application. The dictionary contains as many of the keys described in Table 172-1 (page 3483) as are available.

Special Considerations

It is strongly suggested that you use the `NSRunningApplication` classes' `currentApplication` (page 2275) or `active`methods to retrieve this information in applications targeted for Mac OS X v10.6 and later.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `runningApplications` (page 3476)
- `launchedApplications` (page 3465)

Related Sample Code

iChatStatusFromApplication

Declared In

`NSWorkspace.h`

checkForRemovableMedia

Polls the system's drives for any disks that have been inserted but not yet mounted. (Deprecated in Mac OS X v10.6.)

```
- (void)checkForRemovableMedia
```

Discussion

This method doesn't wait until such disks are mounted; instead, it requests that the disk be mounted asynchronously and returns immediately. Currently has no effect.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- `mountNewRemovableMedia` (page 3467)
- `mountedRemovableMedia` (page 3466)

Declared In

`NSWorkspace.h`

desktopImageOptionsForScreen:

Returns the desktop image options for the given screen.

```
- (NSDictionary *)desktopImageOptionsForScreen:(NSScreen *)screen
```

Parameters

screen

The screen for which to get the desktop image options.

Return Value

A dictionary containing key-value pairs specified in "Desktop Image Dictionary Keys" (page 3487).

Availability
Available in Mac OS X v10.6 and later.

See Also
- `desktopImageURLForScreen:` (page 3455)
- `setDesktopImageURL:forScreen:options:error:` (page 3477)

Related Sample Code
DesktopImage

Declared In
`NSWorkspace.h`

desktopImageURLForScreen:

Returns the URL for the desktop image for the given screen.

```
- (NSURL *)desktopImageURLForScreen:(NSScreen *)screen
```

Parameters

screen
The screen for which to get the desktop image.

Return Value
The desktop image.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `desktopImageOptionsForScreen:` (page 3454)
- `setDesktopImageURL:forScreen:options:error:` (page 3477)

Related Sample Code
AnimatedTableView
DesktopImage

Declared In
`NSWorkspace.h`

duplicateURLs:completionHandler:

Duplicates the specified URLS asynchronously in the same manner as the Finder..

```
- (void)duplicateURLs:(NSArray *)URLs completionHandler:(void (^)(NSDictionary
    *newURLs, NSError *error))handler
```

Parameters

URLs
The array of URLs to duplicate.

handler

The completion handler block object. If completionHandler is not `nil`, it will be called when the operation is complete, on the same dispatch queue that was used for the `duplicateURLs:completionHandler:` call. The completionHandler may be nil if you are not interested in the results.

The block takes two arguments:

newURLs

A dictionary parameter that maps the given URLs to their new URLs locations. Files that could not be duplicated will not be present in the dictionary.

error

If the operation succeeded for every file, the error parameter will be nil. If it failed for one or more files, the error parameter will describe the overall result of the operation in a manner suitable for presentation to the user.

Discussion
This methods may show a progress indicator, or other user interface elements, at AppKit's discretion.

In Mac OS X 10.6, this method require that the main run loop be run in a common mode.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- recycleURLs:completionHandler:` (page 3475)

Declared In
`NSWorkspace.h`

extendPowerOffBy:

Requests the system wait for the specified amount of time before turning off the power or logging out the user.

```
- (NSInteger)extendPowerOffBy:(NSInteger)requested
```

Parameters

requested

The number of milliseconds to wait before turning off the power or logging off the user.

Return Value
The number of milliseconds granted by the system.

Discussion
Currently unimplemented.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWorkspace.h`

fileLabelColors

Returns the corresponding array of file label colors for the file labels.

```
- (NSArray *)fileLabelColors
```

Return Value
An array of `NSColor` objects.

Discussion
This array has the same number of elements as `fileLabels` (page 3457), and the color at a given index corresponds to the label at the same index.

You can listen for notifications named `NSWorkspaceDidChangeFileLabelsNotification` (page 3495) to be notified when file labels change which may result in changes to the order of the `fileLabelColors`.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- fileLabels` (page 3457)

Declared In
`NSWorkspace.h`

fileLabels

Returns the array of file labels as strings.

```
- (NSArray *)fileLabels
```

Return Value
An array of strings.

Discussion
You can listen for notifications named `NSWorkspaceDidChangeFileLabelsNotification` (page 3495) to be notified when file labels change.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- fileLabelColors` (page 3457)

Declared In
`NSWorkspace.h`

filenameExtension:isValidForType:

Returns whether the specified filename extension is appropriate for the Uniform Type Identifier.

```
- (BOOL)filenameExtension:(NSString *)filenameExtension isValidForType:(NSString
    *)typeName
```

Parameters

filenameExtension

A string containing the filename extension.

typeName

A string containing the Uniform Type Identifier.

Return Value

YES if *fileNameExtension* is a valid extension for *typeName*, NO otherwise

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSWorkspace.h

fileSystemChanged

Returns a Boolean value indicating whether a change to the file system has been registered with a noteFileSystemChanged (page 3468) message since the last fileSystemChanged (page 3458) message. (Deprecated in Mac OS X v10.6.)

```
- (BOOL)fileSystemChanged
```

Return Value

Currently, this method always returns NO.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

Declared In

NSWorkspace.h

findApplications

Examines all applications and updates the records of registered services and file types. (Deprecated in Mac OS X v10.6.)

```
- (void)findApplications
```

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

Declared In

NSWorkspace.h

fullPathForApplication:

Returns the full path for the specified application.

```
- (NSString *)fullPathForApplication:(NSString *)appName
```

Parameters

appName

The name of the application.

Return Value

The full path for the application, or `nil` if the specified application was not found.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSWorkspace.h`

getFileSystemInfoForPath:isRemovable:isWritable:isUnmountable:description:type:

Describes the file system at *fullPath*.

```
- (BOOL)getFileSystemInfoForPath:(NSString *)fullPath isRemovable:(BOOL
    *)removableFlag isWritable:(BOOL *)writableFlag isUnmountable:(BOOL
    *)unmountableFlag description:(NSString **)description type:(NSString
    **)fileSystemType
```

Parameters

fullPath

The path to the file-system mount point.

removableFlag

On input, a boolean variable; on return, this variable contains `YES` if the file system is on removable media.

writableFlag

On input, a boolean variable; on return, this variable contains `YES` if the file system writable.

unmountableFlag

On input, a boolean variable; on return, this variable contains `YES` if the file system is unmountable.

description

On input, a pointer to a string object variable; on return, if the method was successful, this variable contains a string object that describes the file system. You should not rely on this description for program logic but can use it in message strings. Values can include “hard,” “nfs,” and “foreign."

fileSystemType

On input, a pointer to a string object variable; on return, if the method was successful, this variable contains the file-system type. Values can include “HFS,” “UFS,” or other values.

Return Value

`YES` if the information was successfully returned, otherwise `NO`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSWorkspace.h`

getInfoForFile:application:type:

Retrieves information about the specified file.

```
- (BOOL)getInfoForFile:(NSString *)fullPath application:(NSString **)appName
    type:(NSString **)type
```

Parameters

fullPath

The full path to the desired file.

appName

The application the system would use to open the file.

type

On input, a pointer to a string object variable; on return, if the method is successful, this variable contains a string object with the filename extension or encoded HFS file type of the file.

Return Value

`YES` if the information was retrieved successfully; otherwise, `NO` if the file could not be found or the application was not associated with the file.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `iconForFile:` (page 3460)
- `iconForFiles:` (page 3461)

Declared In

`NSWorkspace.h`

hideOtherApplications

Hides all applications other than the sender.

```
- (void)hideOtherApplications
```

Discussion

The user can hide all applications except the current one by Command-Option-clicking on an application's Dock icon.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSWorkspace.h`

iconForFile:

Returns an image containing the icon for the specified file.

```
- (NSImage *)iconForFile:(NSString *)fullPath
```

Parameters

fullPath

The full path to the file.

Return Value

The icon associated with the file.

Discussion

The returned image has an initial size of 32 pixels by 32 pixels.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `getInfoForFile:application:type:` (page 3460)
- `iconForFileType:` (page 3462)
- `iconForFiles:` (page 3461)

Related Sample Code

DesktopImage

FunHouse

iChatStatusFromApplication

SourceView

Spotlighter

Declared In

`NSWorkspace.h`

iconForFiles:

Returns an image containing the icon for the specified files.

```
- (NSImage *)iconForFiles:(NSArray *)fullPaths
```

Parameters

fullPaths

An array of `NSString` objects, each of which contains the full path to a file.

Return Value

The icon associated with the group of files.

Discussion

If *fullPaths* specifies one file, that file's icon is returned. If *fullPaths* specifies more than one file, an icon representing the multiple selection is returned.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `iconForFile:` (page 3460)
- `iconForFileType:` (page 3462)

Declared In

`NSWorkspace.h`

iconForFileType:

Returns an image containing the icon for files of the specified type.

```
- (NSImage *)iconForFileType:(NSString *)fileType
```

Parameters

fileType

The file type, which may be either a filename extension or an encoded HFS file type.

Return Value

The icon associated with files of the given type.

Discussion

The returned image has an initial size of 32 pixels by 32 pixels.

Availability

Available in Mac OS X v10.0 and later.

See Also

- iconForFile: (page 3460)
- iconForFiles: (page 3461)

Related Sample Code

ButtonMadness
IconCollection
MyPhoto
SourceView
ZipBrowser

Declared In

NSWorkspace.h

isFilePackageAtPath:

Determines whether the specified path is a file package.

```
- (BOOL)isFilePackageAtPath:(NSString *)fullPath
```

Parameters

fullPath

The full path to examine.

Return Value

YES if the path identifies a file package; otherwise, NO if the path does not exist, is not a directory, or is not a file package.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSWorkspace.h

launchApplication:

Launches the specified application.

```
- (BOOL)launchApplication:(NSString *)appName
```

Parameters

appName

The name of the application to open.

Return Value

YES if the application was successfully launched or was already running; otherwise, NO.

Discussion

The *appName* parameter need not be specified with a full path and, in the case of an application wrapper, may be specified with or without the .app extension, as described in "Use of .app Extension".

Before this method begins, it posts an NSWorkspaceWillLaunchApplicationNotification (page 3491) to the NSWorkspace object's notification center. When the operation is complete, it posts an NSWorkspaceDidLaunchApplicationNotification (page 3491).

Availability

Available in Mac OS X v10.0 and later.

See Also

- launchApplication:showIcon:autolaunch: (page 3463)

Related Sample Code

QTAudioExtractionPanel

Declared In

NSWorkspace.h

launchApplication:showIcon:autolaunch:

Launches the specified application using additional options.

```
- (BOOL)launchApplication:(NSString *)appName showIcon:(BOOL)showIcon
    autolaunch:(BOOL)autolaunch
```

Parameters

appName

The name of the application to open.

showIcon

If NO, the application's icon is not placed on the screen. (The icon still exists, though.)

autolaunch

If YES, the autolaunch default is set as though the specified application were autolaunched at startup.

Return Value

YES if the application was successfully launched or was already running; otherwise, NO.

Discussion

This method is provided to enable daemon-like applications that lack a normal user interface. Its use is not generally encouraged.

Returns `YES` if the application is successfully launched or already running, and `NO` if it can't be launched.

Before this method begins, it posts an `NSWorkspaceWillLaunchApplicationNotification` (page 3491) to the `NSWorkspace` object's notification center. When the operation is complete, it posts an `NSWorkspaceDidLaunchApplicationNotification` (page 3491).

Availability
Available in Mac OS X v10.0 and later.

See Also
`- launchApplication:` (page 3463)

Declared In
`NSWorkspace.h`

launchApplicationAtURL:options:configuration:error:

Launches the app at the specified URL.

```
- (NSRunningApplication *)launchApplicationAtURL:(NSURL
    *)url options:(NSWorkspaceLaunchOptions)options configuration:(NSDictionary
    *)configuration error:(NSError **)error
```

Parameters

url

The application URL.

options

Options to use when launching the application. See "NSWorkspaceLaunchOptions" (page 3487) for possible values.

configuration

A dictionary containing the configuration options. Possible key-value pairs are described in "Workspace Launch Configuration Options" (page 3484)

error

Return Value

Discussion

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSWorkspace.h`

launchAppWithBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifier:

Launches the application corresponding to the specified *bundleIdentifier*.

```
- (BOOL)launchAppWithBundleIdentifier:(NSString *)bundleIdentifier
    options:(NSWorkspaceLaunchOptions)options
    additionalEventParamDescriptor:(NSAppleEventDescriptor *)descriptor
    launchIdentifier:(NSNumber **)identifier
```

Parameters

bundleIdentifier

A bundle identifier string. This value corresponds to the value in the `CFBundleIdentifier` key of the application's `Info.plist` file. For example, the bundle identifier of the TextEdit application is `com.apple.TextEdit`.

options

Options to use when launching the application. Values for this parameter are described in "NSWorkspaceLaunchOptions" (page 3487).

descriptor

Additional options specified in an AppleEvent-style descriptor. For example, you could use this parameter to specify additional documents to open when the application is launched.

identifier

The *launchIdentifiers* are currently unused, and you should pass `NULL`.

Return Value

`YES` if the application was found and launched; otherwise, `NO`.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- openURLs:withAppBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifiers:` (page 3473)

Declared In

`NSWorkspace.h`

launchedApplications

Returns an array of dictionaries, one entry for each running application.

```
- (NSArray *)launchedApplications
```

Return Value

An array of `NSDictionary` objects. Each dictionary contains as many of the keys described in Table 172-1 (page 3483) as are available.

Special Considerations

It is strongly suggested that you use the `NSWorkspace runningApplications` (page 3476) method and the `NSRunningApplication` class to retrieve this information in applications targeted for Mac OS X v10.6 and later.

Availability

Available in Mac OS X v10.2 and later.

See Also

`- runningApplications` (page 3476)

`- activeApplication` (page 3453)

Declared In
NSWorkspace.h

localizedDescriptionForType:

Returns the localized description for the specified Uniform Type Identifier

```
- (NSString *)localizedDescriptionForType:(NSString *)typeName
```

Parameters

typeName
A string containing the Uniform Type Identifier.

Return Value
An NSString containing the localized description of *typeName*.

Discussion
The localized description is suitable for displaying to the user.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
CocoaSlides

Declared In
NSWorkspace.h

mountedLocalVolumePaths

Returns the mount points of all local volumes, not just the removable ones returned by `mountedRemovableMedia` (page 3466).

```
- (NSArray *)mountedLocalVolumePaths
```

Return Value
An array of `NSString` objects, each of which contains the full pathname of the mount point for any local volumes.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
SourceView

Declared In
NSWorkspace.h

mountedRemovableMedia

Returns the full pathnames of all currently mounted removable disks.

```
- (NSArray *)mountedRemovableMedia
```

Return Value
An array of `NSString` objects, each of which contains the full pathname of a mounted removable disk.

Discussion
If the computer provides an interrupt or other notification when the user inserts a disk into a drive, the Finder will mount the disk immediately. However, if no notification is given, the Finder won't be aware that a disk needs to be mounted. On such systems, an application should invoke either `mountNewRemovableMedia` (page 3467) or `checkForRemovableMedia` (page 3454) before invoking `mountedRemovableMedia` (page 3466). Either of these methods cause the Finder to poll the drives to see if a disk is present. If a disk has been inserted but not yet mounted, these methods will cause the Finder to mount it.

The Disk button in an Open or Save panel invokes `mountedRemovableMedia` (page 3466) and `mountNewRemovableMedia` (page 3467) as part of its operation, so most applications won't need to invoke these methods directly.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `checkForRemovableMedia` (page 3454)
- `mountNewRemovableMedia` (page 3467)

Related Sample Code
CocoaDVDPlayer

Declared In
`NSWorkspace.h`

mountNewRemovableMedia

Returns the full pathnames of any newly mounted disks. (Deprecated in Mac OS X v10.6.)

```
- (NSArray *)mountNewRemovableMedia
```

Return Value
An array of NSString objects, each of which contains the full pathname to a newly mounted disk.

Discussion
This method polls the system's drives for any disks that have been inserted but not yet mounted and waits until the new disks have been mounted. This method posts an `NSWorkspaceDidMountNotification` (page 3494) to the `NSWorkspace` object's notification center when it is finished. Currently provides the same functionality as `mountedRemovableMedia` (page 3466).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- `checkForRemovableMedia` (page 3454)
- `mountedRemovableMedia` (page 3466)

Declared In
`NSWorkspace.h`

noteFileSystemChanged

Informs the `NSWorkspace` object that the file system has changed. (Deprecated in Mac OS X v10.6.)

```
- (void)noteFileSystemChanged
```

Discussion
The `NSWorkspace` object then gets the status of all the files and directories it is interested in and updates itself appropriately. This method is used by many objects that write or delete files.

The `NSDocument` and `NSSavePanel` objects use this method when saving a file. If you create a file directly, you should call `noteFileSystemChanged` (page 3468) so that the Finder can update the folder if it is open.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
`- fileSystemChanged` (page 3458)

Declared In
`NSWorkspace.h`

noteFileSystemChanged:

Informs the `NSWorkspace` object that the file system changed at the specified path.

```
- (void)noteFileSystemChanged:(NSString *)path
```

Parameters
path
The full path that changed.

Discussion
The `NSWorkspace` object then gets the status of all the files and directories it is interested in and updates itself appropriately. This method is used by many objects that write or delete files.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- fileSystemChanged` (page 3458)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSWorkspace.h`

noteUserDefaultsChanged

Informs the `NSWorkspace` object that the defaults database has changed. (Deprecated in Mac OS X v10.6.)

```
- (void)noteUserDefaultsChanged
```

Discussion
The `NSWorkspace` object then reads all the defaults it is interested in and reconfigures itself appropriately. For example, this method is used by the Preferences application to notify the Finder whether the user prefers to see hidden files. Currently has no effect.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
`- userDefaultsChanged` (page 3483)

Declared In
`NSWorkspace.h`

notificationCenter

Returns the notification center for workspace notifications.

```
- (NSNotificationCenter *)notificationCenter
```

Return Value
The notification center object.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaDVDPlayer
iChatStatusFromApplication

Declared In
`NSWorkspace.h`

openFile:

Opens the specified file specified using the default application associated with its type.

```
- (BOOL)openFile:(NSString *)fullPath
```

Parameters
fullPath
The full path to the file.

Return Value
`YES` if the file was successfully opened; otherwise, `NO`.

Discussion
The sending application is deactivated before the request is sent.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- openFile:fromImage:at:inView:` (page 3470)
- `- openFile:withApplication:` (page 3471)
- `- openFile:withApplication:andDeactivate:` (page 3471)
- `- openTempFile:` (page 3472)

Related Sample Code
IconCollection
iSpend
QTRecorder
Quartz Composer WWDC 2005 TextEdit
SourceView

Declared In
`NSWorkspace.h`

openFile:fromImage:at:inView:

Opens a file using the default application for its type and animates the action using a custom icon.

```
- (BOOL)openFile:(NSString *)fullPath fromImage:(NSImage *)anImage at:(NSPoint)point
    inView:(NSView *)aView
```

Parameters

fullPath
The full path to the file.

anImage
The icon for the file.

point
The point in `aView` at which to display the icon.

aView
The view in which to display the icon.

Return Value
`YES` if the file was successfully opened; otherwise, `NO`.

Discussion
The Finder provides an animation before opening the file to give the user feedback that the file is to be opened. To provide this animation, *anImage* should contain an icon for the file, and its image should be displayed at *point*, specified in the coordinates of *aView*. Currently provides the same functionality as `openFile:` (page 3469).

The sending application is deactivated before the request is sent.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- openFile:` (page 3469)
- `- openFile:withApplication:` (page 3471)
- `- openFile:withApplication:andDeactivate:` (page 3471)

- openTempFile: (page 3472)

Declared In
NSWorkspace.h

openFile:withApplication:

Opens a file using the specified application.

```
- (BOOL)openFile:(NSString *)fullPath withApplication:(NSString *)appName
```

Parameters

fullPath
The full path to the file.

appName
The name of the application to use when opening the file.

Return Value
YES if the file was successfully opened; otherwise, NO.

Discussion
The *appName* parameter need not be specified with a full path and, in the case of an application wrapper, may be specified with or without the .app extension, as described in "Use of .app Extension". The sending application is deactivated before the request is sent.

Availability
Available in Mac OS X v10.0 and later.

See Also
- openFile: (page 3469)
- openFile:withApplication:andDeactivate: (page 3471)

Related Sample Code
CIVideoDemoGL
ExtractMovieAudioToAIFF
QTExtractAndConvertToAIFF
Quartz Composer WWDC 2005 TextEdit
WhackedTV

Declared In
NSWorkspace.h

openFile:withApplication:andDeactivate:

Opens the specified file and optionally deactivates the sending application.

```
- (BOOL)openFile:(NSString *)fullPath withApplication:(NSString *)appName
    andDeactivate:(BOOL)flag
```

Parameters

fullPath
The full path to the file.

appName
The name of the application to use when opening the file.

flag
If YES, the sending application is deactivated before the request is sent, allowing the opening application to become the active application.

Return Value
YES if the file was successfully opened; otherwise, NO.

Discussion
The *appName* parameter need not be specified with a full path and, in the case of an application wrapper, may be specified with or without the .app extension, as described in "Use of .app Extension". If *appName* is nil, the default application for the file's type is used.

Availability
Available in Mac OS X v10.0 and later.

See Also
- openFile: (page 3469)
- openFile:withApplication: (page 3471)
application:openFile: (page 3566) (NSApplicationDelegate)

Related Sample Code
Core Data HTML Store
DispatchFractal

Declared In
NSWorkspace.h

openTempFile:

Opens the specified temporary file using the default application for its type. (Deprecated in Mac OS X v10.6.)

```
- (BOOL)openTempFile:(NSString *)fullPath
```

Parameters
fullPath
The full path to the temporary file.

Return Value
YES if the file was successfully opened; otherwise, NO.

Discussion
The sending application is deactivated before the request is sent. Using this method instead of one of the openFile:... methods lets the receiving application know that it should delete the file when it no longer needs it. Currently provides the same functionality as openFile: (page 3469).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

See Also
- openFile: (page 3469)
- openFile:fromImage:at:inView: (page 3470)

- openFile:withApplication: (page 3471)
- openFile:withApplication:andDeactivate: (page 3471)

Declared In
NSWorkspace.h

openURL:

Opens the location at the specified URL.

```
- (BOOL)openURL:(NSURL *)url
```

Parameters

url

A URL specifying the location to open.

Return Value
YES if the location was successfully opened; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ObjectPath
PhotoSearch
QTCompressionOptionsWindow
UIElementInspector
VertexPerformanceTest

Declared In
NSWorkspace.h

openURLs:withAppBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifiers:

Opens one or more files from an array of URLs.

```
- (BOOL)openURLs:(NSArray *)urls withAppBundleIdentifier:(NSString *)bundleIdentifier
    options:(NSWorkspaceLaunchOptions)options
   additionalEventParamDescriptor:(NSAppleEventDescriptor *)descriptor
   launchIdentifiers:(NSArray **)identifiers
```

Parameters

urls

An array of NSURL objects, each one identifying a URL for the application to open.

bundleIdentifier

A bundle identifier string or nil to use the default system bindings. This value corresponds to the value in the CFBundleIdentifier key of the application's Info.plist file. For example, the bundle identifier of the TextEdit application is com.apple.TextEdit.

options

Options to use when launching the application. Values for this parameter are described in "NSWorkspaceLaunchOptions" (page 3487).

descriptor

Additional options specified in an AppleEvent-style descriptor. For example, you could use this parameter to specify additional documents to open when the application is launched.

identifiers

The *launchIdentifiers* are currently unused, and you should pass `NULL`.

Return Value
`YES` if the application was found and launched; otherwise, `NO`.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- launchAppWithBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifier:` (page 3464)

Related Sample Code
NewsReader

Declared In
`NSWorkspace.h`

performFileOperation:source:destination:files:tag:

Performs a file operation on a set of files in a particular directory.

```
- (BOOL)performFileOperation:(NSString *)operation source:(NSString *)source
    destination:(NSString *)destination files:(NSArray *)files tag:(NSInteger *)tag
```

Parameters

operation

The file operation to perform. The possible values for this parameter are described in "File Operations" (page 3485).

source

The full path to the directory containing the files on which to operate.

destination

The full path to the destination directory of the operation.

files

An array of `NSString` objects specifying the names of the files and directories to be manipulated. Each string must not contain any path information other than the name of the file or directory. In other words, all of the files and directories must be located in the source directory and not in one if its subdirectories.

tag

On input, a integer variable; on return, this variable contains a negative integer if the operation fails, 0 if the operation was performed synchronously and succeeded, or a positive integer if the operation was performed asynchronously. If the value is a positive integer, the value is a tag that identifies the requested file operation.

Return Value
`YES` if the operation succeeded; otherwise, `NO`.

Discussion
Some operations—such as moving, copying, and linking files—require a destination directory to be specified. If not, *`destination`* should be the empty string (`@""`). Before this method returns, it posts an `NSWorkspaceDidPerformFileOperationNotification` (page 3494) to the `NSWorkspace` object's notification center.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSWorkspace.h`

preferredFilenameExtensionForType:

Returns the preferred filename extension for the specified Uniform Type Identifier.

```
- (NSString *)preferredFilenameExtensionForType:(NSString *)typeName
```

Parameters
`typeName`
A string containing the Uniform Type Identifier.

Return Value
The appropriate filename extension for typeName, or `nil` if no extension could be determined.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSWorkspace.h`

recycleURLs:completionHandler:

Moves the specified URLs to the trash in the same manner a the Finder.

```
- (void)recycleURLs:(NSArray *)URLs completionHandler:(void (^)(NSDictionary
    *newURLs, NSError *error))handler
```

Parameters
`URLs`
The array of URLs to move to the trash.

handler

The completion handler block object. If completionHandler is not `nil`, it will be called when the operation is complete, on the same dispatch queue that was used for the `recycleURLs:completionHandler:` call. The completionHandler may be nil if you are not interested in the results.

The block takes two arguments:

newURLs

A dictionary parameter that maps the given URLs to their new URLs locations in the trash. Files that could not be moved to the trash will not be present in the dictionary.

error

If the operation succeeded for every file, the error parameter will be nil. If it failed for one or more files, the error parameter will describe the overall result of the operation in a manner suitable for presentation to the user.

Discussion
This methods may show a progress indicator, or other user interface elements, at AppKit's discretion.

In Mac OS X 10.6, this method require that the main run loop be run in a common mode.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- duplicateURLs:completionHandler:` (page 3455)

Declared In
`NSWorkspace.h`

runningApplications

Returns an array of `NSRunningApplication` representing the running applications.

```
- (NSArray *)runningApplications
```

Return Value
An array of `NSRunningApplication` instances.

Discussion
The order of the array is unspecified, but it is stable, meaning that the relative order of particular applications will not change across multiple calls to `runningApplications`. See *NSRunningApplication Class Reference* for more information on `NSRunningApplication`.

Similar to the `NSRunningApplication` class's properties, this property will only change when the main run loop is run in a common mode. Instead of polling, use key-value observing to be notified of changes to this array property.

This property is thread safe, in that it may be called from background threads and the result is returned atomically.

This property is observable using key-value observing.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSRunningApplication.h

selectFile:inFileViewerRootedAtPath:

Selects the file specified by *fullPath*.

```
- (BOOL)selectFile:(NSString *)fullPath inFileViewerRootedAtPath:(NSString
    *)rootFullPath
```

Parameters

fullPath
The full path of the file to select.

rootFullPath
If a path is specified, a new file viewer is opened. If you specify an empty string (@"") for this parameter, the file is selected in the main viewer.

Return Value
YES if the file was successfully selected; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
EnhancedAudioBurn

Declared In
NSWorkspace.h

setDesktopImageURL:forScreen:options:error:

Sets the desktop image for the given screen to the image at the specified URL.

```
- (BOOL)setDesktopImageURL:(NSURL *)url forScreen:(NSScreen *)screen
    options:(NSDictionary *)options error:(NSError **)error
```

Parameters

url
A file URL to the image. The URL must not be nil.

screen
The screen to set the desktop image on.

options
The options dictionary may contain any of the "Desktop Image Dictionary Keys" (page 3487) keys, which control how the image is scaled on the screen.

error
A error that is returned by-reference if setting the image fails.

Return Value
`YES` if the image was set as the desktop, otherwise `NO`. If `NO` is returned, the *`error`* parameter provides additional information.

Discussion
You should not present a user interface for picking the options. Instead, choose appropriate defaults and allow the user to adjust them in the System Preference Pane.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- desktopImageOptionsForScreen:` (page 3454)
`- desktopImageURLForScreen:` (page 3455)

Related Sample Code
AnimatedTableView
DesktopImage

Declared In
`NSWorkspace.h`

setIcon:forFile:options:

Sets the icon for the file or directory at the specified path.

```
- (BOOL)setIcon:(NSImage *)image forFile:(NSString *)fullPath
    options:(NSWorkspaceIconCreationOptions)options
```

Parameters

`image`
The image to use as the icon for the file or directory.

`fullPath`
The full path of the file or directory.

`options`
The icon representations to generate from the image. You specify this value by combining the appropriate "Workspace icon creation options" (page 3490) constants, using the C bitwise `OR` operator. Specify `0` if you want to generate icons in all available icon representation formats.

Return Value
`YES` if the icon was set; otherwise, `NO`.

Discussion
The *`image`* can be an arbitrary image, with or without transparency. This image is automatically scaled (as needed) to generate the icon representations. The file or folder must exist and be writable by the user.

It is recommended that applications include the `NSExclude10_4ElementsIconCreationOption` option for compatibility with pre-Mac OS X v10.3 Finder. Icons that include the high resolution elements prevent custom icons from being displayed on earlier systems.

Availability
Available in Mac OS X v10.4 and later.

Declared In
NSWorkspace.h

showSearchResultsForQueryString:

Displays a Spotlight search results window in Finder for the specified query string.

```
- (BOOL)showSearchResultsForQueryString:(NSString *)queryString
```

Parameters

queryString
The string to search for.

Return Value
YES if the communication with Finder was successful, otherwise NO.

Discussion
Finder becomes the active application, if possible. The user can further refine the search via the Finder user interface.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSWorkspace.h

slideImage:from:to:

Animates a sliding image from one point to another. (Deprecated in Mac OS X v10.6.)

```
- (void)slideImage:(NSImage *)image from:(NSPoint)fromPoint to:(NSPoint)toPoint
```

Parameters

image
The image to animate.

fromPoint
The starting point, in screen coordinates.

toPoint
The ending point, in screen coordinates.

Discussion
Currently unimplemented.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

Declared In
NSWorkspace.h

type:conformsToType:

Returns a Boolean indicating that the first Uniform Type Identifier conforms to the second Uniform Type Identifier.

```
- (BOOL)type:(NSString *)firstTypeName conformsToType:(NSString *)secondTypeName
```

Parameters

firstTypeName

A string containing the Uniform Type Identifier that should conform to *secondTypeName*.

secondTypeName

A string containing a Uniform Type Identifier.

Return Value

YES if *firstTypeName* conforms to the uniform type identifier hierarchy of *secondTypeName*, NO otherwise.

Discussion

Use this method instead of comparing Uniform Identifier Types for equality. See *Uniform Type Identifiers Overview* for information about Uniform Type Identifier conformance.

This method will always return YES if the two strings are equal. It is appropriate to use this method with other type names, including those declared in CFBundleTypeName Info.plist entries.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSWorkspace.h

typeOfFile:error:

Returns the uniform type identifier of the specified file, if it can be determined..

```
- (NSString *)typeOfFile:(NSString *)absoluteFilePath error:(NSError **)outError
```

Parameters

absoluteFilePath

The absolute path of the file.

outError

If the Uniform Type Identifier of the file at absolutePath can't be determined, *outError* contains an NSError object that describes why.

Return Value

An NSString containing the uniform type identifier of the file at absoluteFilePath. If no UTI can be determined the return value is nil.

Discussion

If the file at the specified path is a symbolic link, the type of the symbolic link is returned.

Availability

Available in Mac OS X v10.5 and later.

Related Sample Code

CocoaSlides

Quartz 2D Transformer

Declared In
NSWorkspace.h

unmountAndEjectDeviceAtPath:

Unmounts and ejects the device at the specified path.

```
- (BOOL)unmountAndEjectDeviceAtPath:(NSString *)path
```

Parameters

path
The path to the device.

Return Value
YES if the device was unmounted; otherwise, NO.

Discussion
When this method begins, it posts an NSWorkspaceWillUnmountNotification (page 3494) to the NSWorkspace object's notification center. When it is finished, it posts an NSWorkspaceDidUnmountNotification (page 3494).

The unmountAndEjectDeviceAtURL:error: (page 3481) is preferable as it will provide more detailed error information.

Availability
Available in Mac OS X v10.0 and later.

See Also
- unmountAndEjectDeviceAtURL:error: (page 3481)

Related Sample Code
CocoaDVDPlayer

Declared In
NSWorkspace.h

unmountAndEjectDeviceAtURL:error:

Attempts to eject the volume mounted at the given path.

```
- (BOOL)unmountAndEjectDeviceAtURL:(NSURL *)url error:(NSError **)error
```

Parameters

url
The URL of the volume to eject.

error
If the operation fails, this error contains more information about the failure.

Return Value
YES if the volume was unmounted and ejected successfully, otherwise NO, for example, if the volume is not ejectable.

Availability
Available in Mac OS X v10.6 and later.

See Also
`- unmountAndEjectDeviceAtPath:` (page 3481)

Declared In
`NSWorkspace.h`

URLForApplicationToOpenURL:

Returns the URL to the default application that would be used to open the given URL.

```
- (NSURL *)URLForApplicationToOpenURL:(NSURL *)url
```

Parameters

url
The URL of the file to open.

Return Value
The URL of the default application that would open the specified *url*. Returns nil if no application is able to open the url, or if the file url does not exist.

Discussion
This is the programmatic equivalent of double clicking a document in the Finder.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSWorkspace.h`

URLForApplicationWithBundleIdentifier:

Returns the URL for the application with the specified identifier.

```
- (NSURL *)URLForApplicationWithBundleIdentifier:(NSString *)bundleIdentifier
```

Parameters

bundleIdentifier
A bundle identifier specifying an application.

Return Value
The URL of the application, or `nil` if no application has the bundle identifier.

Discussion
This uses various (currently unspecified) heuristics in case multiple apps have the same bundle ID.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSWorkspace.h`

userDefaultsChanged

Returns a Boolean value indicating whether a change to the defaults database has been registered with a `noteUserDefaultsChanged` (page 3468) message since the last `userDefaultsChanged` (page 3483) message. (Deprecated in Mac OS X v10.6.)

```
- (BOOL)userDefaultsChanged
```

Return Value
Currently, this method always returns `NO`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.

Declared In
`NSWorkspace.h`

Constants

The following table describes keys for an `NSDictionary` object containing information about an application. This dictionary is returned by `activeApplication` (page 3453) and `launchedApplications` (page 3465), and is also provided in the *`userInfo`* of `NSWorkspace` notifications for application launch and termination.

Note that these constants are considered legacy.

Note: It is strongly suggested that you use the `NSWorkspace`class's `runningApplications` (page 3476) method and the `NSRunningApplication` class to retrieve this information in applications targeted to Mac OS X v10.6 and later, rather than the `activeApplication` (page 3453) and `launchedApplications` (page 3465) methods.

Table 172-1 `userInfo` dictionary keys for `activeApplication` and `launchedApplications` and notifications for application launch and termination.

Key	Value
`@"NSApplicationPath"`	The full path to the application, as a `NSString` object.
`@"NSApplicationName"`	The application's name, as an `NSString` object.
`@"NSApplicationBundleIdentifier"`	The application's bundle identifier, as an `NSString` object.
`@"NSApplicationProcessIdentifier"`	The application's process id, as an `NSNumber` object.
`@"NSApplicationProcessSerialNumberHigh"`	The high long of the process serial number (PSN), as an `NSNumber` object.
`@"NSApplicationProcessSerialNumberLow"`	The low long of the process serial number (PSN), as an `NSNumber` object.

File Types

These constants specify different types of files returned by the `getInfoForFile:application:type:` (page 3460) method.

```
NSString *NSPlainFileType;
NSString *NSDirectoryFileType;
NSString *NSApplicationFileType;
NSString *NSFilesystemFileType;
NSString *NSShellCommandFileType;
```

Constants

`NSPlainFileType`
Plain (untyped) file
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.
Declared in `NSWorkspace.h`.

`NSDirectoryFileType`
Directory
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.
Declared in `NSWorkspace.h`.

`NSApplicationFileType`
Cocoa application
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.
Declared in `NSWorkspace.h`.

`NSFilesystemFileType`
File-system mount point
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.
Declared in `NSWorkspace.h`.

`NSShellCommandFileType`
Executable shell command
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.6.
Declared in `NSWorkspace.h`.

Declared In
`NSWorkspace.h`

Workspace Launch Configuration Options

The following keys can be used in the configuration dictionary of the `launchApplicationAtURL:options:configuration:error:` (page 3464) method. Each key is optional, and if omitted, default behavior is applied.

```
NSString * const NSWorkspaceLaunchConfigurationAppleEvent;
NSString * const NSWorkspaceLaunchConfigurationArguments;
NSString * const NSWorkspaceLaunchConfigurationEnvironment;
NSString * const NSWorkspaceLaunchConfigurationArchitecture;
```

Constants

`NSWorkspaceLaunchConfigurationAppleEvent`

The value is the first `NSAppleEventDescriptor` to send to the new application. If an instance of the application is already running, this is sent to that application.

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchConfigurationArguments`

The value is an `NSArray` of `NSStrings`, passed to the new application in the `argv` parameter. Ignored if a new instance of the application is not launched.

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchConfigurationEnvironment`

The value is an `NSDictionary`, mapping `NSStrings` to `NSStrings`, containing environment variables to set for the new app. Ignored if a new instance of the application is not launched.

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchConfigurationArchitecture`

The value is an NSNumber containing an `Mach-O Architecture` constant. Ignored if a new instance of the application is not launched.

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

File Operations

These constants specify different types of file operations used by `performFileOperation:source:destination:files:tag:` (page 3474).

```
NSString *NSWorkspaceMoveOperation;
NSString *NSWorkspaceCopyOperation;
NSString *NSWorkspaceLinkOperation;
NSString *NSWorkspaceCompressOperation;
NSString *NSWorkspaceDecompressOperation;
NSString *NSWorkspaceEncryptOperation;
NSString *NSWorkspaceDecryptOperation;
NSString *NSWorkspaceDestroyOperation;
NSString *NSWorkspaceRecycleOperation;
NSString *NSWorkspaceDuplicateOperation;
```

Constants

`NSWorkspaceMoveOperation`

Move file to destination. Behaves the same as `movePath:toPath:handler:`.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceCopyOperation`
Copy file to destination. Behaves the same as `copyPath:toPath:handler:`.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLinkOperation`
Create hard link to file in destination. Behaves the same as `linkPath:toPath:handler:`.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceCompressOperation`
Compress file. This operation always returns an error.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceDecompressOperation`
Decompress file. This operation always returns an error.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceEncryptOperation`
Encrypt file. This operation always returns an error.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceDecryptOperation`
Decrypt file. This operation always returns an error.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceDestroyOperation`
Destroy file. Behaves the same as `removeFileAtPath:handler:`.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceRecycleOperation`
Move file to trash. The file is moved to the trash folder on the volume containing the file using the same semantics as `NSWorkspaceMoveOperation`. If a file with the same name currently exists in the trash folder, the new file is renamed. If no trash folder exists on the volume containing the file, the operation fails.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceDuplicateOperation`
Duplicate file in source directory.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

Declared In
`NSWorkspace.h`

Desktop Image Dictionary Keys

The following keys may be specified or returned in the options dictionary for `setDesktopImageURL:forScreen:options:error:` (page 3477).

```
NSString * const NSWorkspaceDesktopImageScalingKey;
NSString * const NSWorkspaceDesktopImageAllowClippingKey;
NSString * const NSWorkspaceDesktopImageFillColorKey;
```

Constants

`NSWorkspaceDesktopImageScalingKey`

The value is an `NSNumber` containing an `NSImageScaling` (page 617) constant as declared in `NSCell`. If this is not specified, `NSImageScaleProportionallyUpOrDown` (page 617) is used. `NSImageScaleProportionallyDown` (page 617) is not currently supported.

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceDesktopImageAllowClippingKey`

The value is an `NSNumber` containing a `BOOL`, which affects the interpretation of Proportional scaling types. A `NO` value will make the image fully visible, but there may be empty space on the sides or top and bottom. A `YES` value will cause the image to fill the entire screen, but the image may be clipped. If this is not specified, `NO` is assumed. Non-proportional scaling types ignore this value.

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceDesktopImageFillColorKey`

The value is an `NSColor`, which is used to fill any empty space around the image. If not specified, a default value is used. Currently, only colors that use or can be converted to use `NSCalibratedRGBColorSpace` are supported, and any alpha value is ignored.

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

NSWorkspaceLaunchOptions

These constants define launch options you can pass to `launchAppWithBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifier:` (page 3464) and `openURLs:withAppBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifiers:` (page 3473).

```
enum {
     NSWorkspaceLaunchAndPrint = 0x00000002,
     NSWorkspaceLaunchInhibitingBackgroundOnly = 0x00000080,
     NSWorkspaceLaunchWithoutAddingToRecents = 0x00000100,
     NSWorkspaceLaunchWithoutActivation = 0x00000200,
     NSWorkspaceLaunchAsync = 0x00010000,
     NSWorkspaceLaunchAllowingClassicStartup = 0x00020000,
     NSWorkspaceLaunchPreferringClassic = 0x00040000,
     NSWorkspaceLaunchNewInstance = 0x00080000,
     NSWorkspaceLaunchAndHide = 0x00100000,
     NSWorkspaceLaunchAndHideOthers = 0x00200000,
     NSWorkspaceLaunchDefault = NSWorkspaceLaunchAsync |
NSWorkspaceLaunchAllowingClassicStartup
};
typedef NSUInteger NSWorkspaceLaunchOptions;
```

Constants

`NSWorkspaceLaunchAndPrint`
Print items instead of opening them.
Available in Mac OS X v10.3 and later.
Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchInhibitingBackgroundOnly`
Causes launch to fail if the target is background-only.
Available in Mac OS X v10.3 and later.
Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchWithoutAddingToRecents`
Do not add the application or documents to the Recents menu.
Available in Mac OS X v10.3 and later.
Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchWithoutActivation`
Launch the application but do not bring it into the foreground.
Available in Mac OS X v10.3 and later.
Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchAsync`
Launch the application and return the results asynchronously.
Available in Mac OS X v10.3 and later.
Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchAllowingClassicStartup`
Start up the Classic compatibility environment, if it is required by the application.
Available in Mac OS X v10.3 and later.
Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchPreferringClassic`
Force the application to launch in the Classic compatibility environment.
Available in Mac OS X v10.3 and later.
Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchNewInstance`
Create a new instance of the application, even if one is already running.

Available in Mac OS X v10.3 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchAndHide`
Tell the application to hide itself as soon as it has finished launching.

Available in Mac OS X v10.3 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchAndHideOthers`
Hide all applications except the newly launched one.

Available in Mac OS X v10.3 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchDefault`
Launch the application asynchronously and launch it in the Classic environment, if required.

Available in Mac OS X v10.3 and later.

Declared in `NSWorkspace.h`.

Volume Mounting Notification User Info Keys

The following keys are available in the `userInfo` parameter of the notification named `NSWorkspaceDidRenameVolumeNotification` (page 3493).

```
NSString * const NSWorkspaceVolumeLocalizedNameKey;
NSString * const NSWorkspaceVolumeURLKey;
```

Constants

`NSWorkspaceVolumeLocalizedNameKey`
NSString containing the user-visible name of the volume.

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceVolumeURLKey`
`NSURL` containing the mount path of the volume.

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

NSWorkspaceDidRenameVolumeNotification User Info Keys

The following keys are available in the `userInfo` parameter of the notification named `NSWorkspaceDidRenameVolumeNotification` (page 3493).

```
NSString * const NSWorkspaceVolumeOldLocalizedNameKey;
NSString * const NSWorkspaceVolumeOldURLKey;
```

Constants

`NSWorkspaceVolumeOldLocalizedNameKey`
`NSString` containing the old user-visible name of the volume

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceVolumeOldURLKey`
`NSURL` containing the old mount path of the volume

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

NSWorkspaceApplicationKey User Info Key

This constant is supplied in the userInfo dictionary of various notifications.

```
NSString * const NSWorkspaceApplicationKey;
```

Constants

`NSWorkspaceApplicationKey`
The value corresponding to this key is an instance of `NSRunningApplication` that reflects the affected application.

Available in Mac OS X v10.6 and later.

Declared in `NSWorkspace.h`.

Workspace icon creation options

These constants describe the `NSWorkspaceIconCreationOptions` values used by `setIcon:forFile:options:` (page 3478). You can combine these using the C bitwise OR operator.

```
enum {
    NSExcludeQuickDrawElementsIconCreationOption    = 1 << 1,
    NSExclude10_4ElementsIconCreationOption        = 1 << 2
};
typedef NSUInteger NSWorkspaceIconCreationOptions;
```

Constants

`NSExcludeQuickDrawElementsIconCreationOption`
Suppress generation of the QuickDraw format icon representations that are used in Mac OS X v10.0 through Mac OS X v10.4.

Available in Mac OS X v10.4 and later.

Declared in `NSWorkspace.h`.

`NSExclude10_4ElementsIconCreationOption`
Suppress generation of the new higher resolution icon representations that are supported in Mac OS X v10.4.

Available in Mac OS X v10.4 and later.

Declared in `NSWorkspace.h`.

Notifications

All `NSWorkspace` notifications are posted to the `NSWorkspace` object's own notification center, not the application's default notification center. Access this center using the `NSWorkspace` object's `notificationCenter` (page 3469) method.

NSWorkspaceWillLaunchApplicationNotification

Posted when the Finder is about to launch an application.

The notification object is the shared `NSWorkspace` instance. In Mac OS X v10.6 and later the *`userInfo`* dictionary contains the `NSWorkspaceApplicationKey` (page 3490) key with a corresponding instance of `NSRunningApplication` that represents the affected application.

In Mac OS X v10.5 and earlier the *`userInfo`* dictionary contains the keys and values described in Table 172-1 (page 3483).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceDidLaunchApplicationNotification

Posted when a new application has started up.

The notification object is the shared `NSWorkspace` instance. In Mac OS X v10.6 and later the *`userInfo`* dictionary contains the `NSWorkspaceApplicationKey` (page 3490) key with a corresponding instance of `NSRunningApplication` that represents the affected application.

In Mac OS X v10.5 and earlier the *`userInfo`* dictionary contains the keys and values described in Table 172-1 (page 3483).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceDidTerminateApplicationNotification

Posted when an application finishes executing.

The notification object is the shared `NSWorkspace` instance. In Mac OS X v10.6 and later the *`userInfo`* dictionary contains the `NSWorkspaceApplicationKey` (page 3490) key with a corresponding instance of `NSRunningApplication` that represents the affected application.

In Mac OS X v10.5 and earlier the *`userInfo`* dictionary contains the keys and values described in Table 172-1 (page 3483).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWorkspace.h

NSWorkspaceSessionDidBecomeActiveNotification

Posted after a user session is switched in. This allows an application to re-enable some processing when a switched out session gets switched back in, for example.

The notification object is the shared NSWorkspace instance. The notification does not contain a *userInfo* dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSWorkspace.h

NSWorkspaceSessionDidResignActiveNotification

Posted before a user session is switched out. This allows an application to disable some processing when its user session is switched out, and re-enable when that session gets switched back in, for example.

The notification object is the shared NSWorkspace instance. The notification does not contain a *userInfo* dictionary.

If an application is launched in an inactive session, NSWorkspaceSessionDidResignActiveNotification is sent after NSApplicationWillFinishLaunchingNotification (page 195) and before sending NSApplicationDidFinishLaunchingNotification (page 194).

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSWorkspace.h

NSWorkspaceDidHideApplicationNotification

Posted when the Finder hid an application.

The notification object is the shared NSWorkspace instance. In Mac OS X v10.6 and later the *userInfo* dictionary contains the NSWorkspaceApplicationKey (page 3490) key with a corresponding instance of NSRunningApplication that represents the affected application.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSWorkspace.h

NSWorkspaceDidUnhideApplicationNotification

Posted when the Finder unhid an application.

The notification object is the shared `NSWorkspace` instance. In Mac OS X v10.6 and later the *`userInfo`* dictionary contains the `NSWorkspaceApplicationKey` (page 3490) key with a corresponding instance of `NSRunningApplication` that represents the affected application.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceDidActivateApplicationNotification

Posted when the Finder is about to activate an application.

The notification object is the shared `NSWorkspace` instance. In Mac OS X v10.6 and later the *`userInfo`* dictionary contains the `NSWorkspaceApplicationKey` (page 3490) key with a corresponding instance of `NSRunningApplication` that represents the affected application.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceDidDeactivateApplicationNotification

Posted when the Finder deactivated an application.

The notification object is the shared `NSWorkspace` instance. In Mac OS X v10.6 and later the *`userInfo`* dictionary contains the `NSWorkspaceApplicationKey` (page 3490) key with a corresponding instance of `NSRunningApplication` that represents the affected application.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceDidRenameVolumeNotification

Posted when a volume changes its name and/or mount path. These typically change simultaneously, in which case only one notification is posted.

The notification object is the shared `NSWorkspace` instance. The `userInfo` dictionary contains keys in "NSWorkspaceDidRenameVolumeNotification User Info Keys" (page 3489) and "Volume Mounting Notification User Info Keys" (page 3489).

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceDidMountNotification

Posted when a new device has been mounted.

The notification object is the shared `NSWorkspace` instance.

In Mac OS X v10.5 and earlier the *`userInfo`* dictionary contains a key `@"NSDevicePath"` that returns the path where the device was mounted, as a string.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceWillUnmountNotification

Posted when the Finder is about to unmount a device.

This notification will not be delivered if a volume was forcibly and immediately made unavailable, such as when a FireWire drive is simply unplugged, because there is no chance to deliver it before the volume becomes unavailable.

The notification object is the shared `NSWorkspace` instance. The *`userInfo`* dictionary contains a key `@"NSDevicePath"` that returns the path where the device was mounted, as a string.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceDidUnmountNotification

Posted when the Finder did unmount a device.

This notification is delivered even if a volume was forcibly and immediately made unavailable, such as when a drive is simply unplugged.

The notification object is the shared `NSWorkspace` instance. The *`userInfo`* dictionary contains a key `@"NSDevicePath"` that returns the path where the device was mounted, as a string.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceDidPerformFileOperationNotification

Posted when a file operation has been performed in the receiving application.

The notification object is the shared `NSWorkspace` instance. The *`userInfo`* dictionary contains a key `@"NSOperationNumber"` with a `NSNumber` object containing an integer indicating the type of file operation

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceDidChangeFileLabelsNotification

Posted when the Finder file labels or colors change.

The notification object is the shared `NSWorkspace` instance. The notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceActiveSpaceDidChangeNotification

Posted when a Spaces change has occurred.

The notification object is the shared `NSWorkspace` instance. The notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceDidWakeNotification

Posted when the machine wakes from sleep.

The notification object is the shared `NSWorkspace` instance. The notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceWillPowerOffNotification

Posted when the user has requested a logout or that the machine be powered off.

The notification object is the shared `NSWorkspace` instance. This notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWorkspace.h

NSWorkspaceWillSleepNotification

Posted before the machine goes to sleep. An observer of this message can delay sleep for up to 30 seconds while handling this notification.

The notification object is the shared `NSWorkspace` instance. The notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSWorkspace.h

NSWorkspaceScreensDidSleepNotification

Posted when the machine's screen goes to sleep.

The notification object is the shared `NSWorkspace` instance. The notification does not contain a *`userInfo`* dictionary.

Few applications are likely to be interested in this notification, but they may be useful for certain hardware-based drawing decisions, for example when using OpenGL.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSWorkspace.h

NSWorkspaceScreensDidWakeNotification

Posted when the machine's screens wake.

The notification object is the shared `NSWorkspace` instance. The notification does not contain a *`userInfo`* dictionary.

Few applications are likely to be interested in this notification, but they may be useful for certain hardware-based drawing decisions, for example when using OpenGL.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSWorkspace.h

Protocols

NSAccessibility Protocol Reference

(informal protocol)

Adopted by	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSAccessibility.h AppKit/NSErrors.h
Companion guide	Accessibility Programming Guidelines for Cocoa

Overview

The `NSAccessibility` informal protocol defines methods that Cocoa classes must implement to make themselves available to an external assistive application. An assistive application interacts with your application to allow persons with disabilities to use your application. For example, a person with a visual impairment could use an application to convert menu items and button labels into speech and then perform actions by verbal command.

Because many Cocoa user interface classes already implement the `NSAccessibility` protocol, providing reasonable default behavior in most cases, Cocoa applications built with standard objects are automatically accessible. In general, you need to explicitly implement the `NSAccessibility` protocol methods only if you subclass one of them, adding new behavior.

The Cocoa implementations of these methods raise an `NSAccessibilityException` when errors occur, such as requesting the value of an unsupported attribute. In most cases, subclasses do not need to raise or catch these exceptions, because overridden methods should invoke their inherited methods for unrecognized attribute and action names.

An accessible object is described by a set of attributes that define characteristics such as the object type, its value, its size and position on the screen, and its place in the accessibility hierarchy. For some objects, the set of attributes can include parameterized attributes. Parameterized attributes behave similar to a function by allowing you to pass a parameter when requesting an attribute value.

See "Accessibility" (page 3951) in *Application Kit Functions Reference* for functions related to accessibility.

Tasks

Accessing Attributes

- `accessibilityAttributeNames` (page 3503)
  Returns an array of attribute names supported by the receiver.
- `accessibilityAttributeValue:` (page 3503)
  Returns the value of the specified attribute in the receiver.
- `accessibilityIsAttributeSettable:` (page 3506)
  Returns a Boolean value that indicates whether the value for the specified attribute in the receiver can be set.
- `accessibilitySetValue:forAttribute:` (page 3508)
  Sets the value of the specified attribute in the receiver to the specified value.
- `accessibilitySetOverrideValue:forAttribute:` (page 3508)
  Overrides the specified attribute in the receiver, or adds it if it does not exist, and sets its value to the specified value.
- `accessibilityArrayAttributeCount:` (page 3502)
  Returns the count of the specified accessibility array attribute.
- `accessibilityArrayAttributeValues:index:maxCount:` (page 3502)
  Returns a subarray of values of an accessibility array attribute.
- `accessibilityIndexOfChild:` (page 3505)
  Returns the index of the specified accessibility child in the parent.

Accessing Parameterized Attributes

- `accessibilityParameterizedAttributeNames` (page 3507)
  Returns a list of parameterized attribute names supported by the receiver.
- `accessibilityAttributeValue:forParameter:` (page 3504)
  Returns the value of the receiver's parameterized attribute corresponding to the specified attribute name and parameter.

Accessing Actions

- `accessibilityActionNames` (page 3501)
  Returns an array of action names supported by the receiver.
- `accessibilityActionDescription:` (page 3501)
  Returns a localized description of the specified action.
- `accessibilityPerformAction:` (page 3507)
  Performs the action associated with the specified action.

Querying Elements

- `accessibilityIsIgnored` (page 3506)
  Returns a Boolean value indicating whether the receiver should be ignored in the parent-child accessibility hierarchy.
- `accessibilityHitTest:` (page 3505)
  Returns the deepest descendant of the accessibility hierarchy that contains the specified point.
- `accessibilityFocusedUIElement` (page 3504)
  Returns the deepest descendant of the accessibility hierarchy that has the focus.

Instance Methods

accessibilityActionDescription:

Returns a localized description of the specified action.

```
- (NSString *)accessibilityActionDescription:(NSString *)action
```

Parameters

action
The action attribute.

Return Value
The description of the specified action, in a localized string.

Discussion
User interface classes must implement this method to return descriptions for all actions returned from `accessibilityActionNames` (page 3501). A button, for example, might return the string "press" for the `NSAccessibilityPressAction` action. Subclasses should invoke the superclass's implementation, if it exists, to obtain the descriptions of any inherited actions.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSAccessibility.h`

accessibilityActionNames

Returns an array of action names supported by the receiver.

```
- (NSArray *)accessibilityActionNames
```

Return Value
An array of action names.

Discussion
User interface classes must implement this method. Subclasses should invoke the superclass's implementation, if it exists, and append additional action names or remove unsupported actions. See "Constants" (page 3509) for some common action names.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
Cocoa Tips and Tricks
ImageMap
ImageMapExample
Sketch+Accessibility
ZipBrowser

Declared In
`NSAccessibility.h`

accessibilityArrayAttributeCount:

Returns the count of the specified accessibility array attribute.

```
- (NSUInteger)accessibilityArrayAttributeCount:(NSString *)attribute
```

Parameters

attribute
The accessibility array attribute.

Return Value
The number of items in the specified array attribute.

Discussion
If `attribute` is not an array an exception is raised.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSAccessibility.h`

accessibilityArrayAttributeValues:index:maxCount:

Returns a subarray of values of an accessibility array attribute.

```
- (NSArray *)accessibilityArrayAttributeValues:(NSString *)attribute
    index:(NSUInteger)index maxCount:(NSUInteger)maxCount
```

Parameters

attribute
The accessibility array attribute.

index
The starting index.

maxCount
The maximum desired number of items requested.

Return Value
An array of values within the specified index and count.

Discussion
Note that this method does not take a range. The max count is the maximum desired number of items requested by an accessibility client. This number may be beyond the bounds of your array.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSAccessibility.h`

accessibilityAttributeNames

Returns an array of attribute names supported by the receiver.

```
- (NSArray *)accessibilityAttributeNames
```

Return Value
An array containing the attributes supported by the receiver.

Discussion
User interface classes must implement this method. Subclasses should invoke the superclass's implementation, if it exists, and append additional attributes or remove unsupported attributes. See "Constants" (page 3509) for lists of attribute names.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
ImageMap
ImageMapExample
Sketch+Accessibility
TrackBall
ZipBrowser

Declared In
`NSAccessibility.h`

accessibilityAttributeValue:

Returns the value of the specified attribute in the receiver.

```
- (id)accessibilityAttributeValue:(NSString *)attribute
```

Parameters
attribute
The name of the attribute. See "Constants" (page 3509) for lists of attribute names.

Discussion
User interface classes must implement this method. Subclasses should invoke the superclass's implementation, if it exists, if *attribute* is not implemented in the subclass.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
ZipBrowser

Declared In
`NSAccessibility.h`

accessibilityAttributeValue:forParameter:

Returns the value of the receiver's parameterized attribute corresponding to the specified attribute name and parameter.

```
- (id)accessibilityAttributeValue:(NSString *)attribute forParameter:(id)parameter
```

Parameters

attribute
The name of the attribute. See "Constants" (page 3509) for lists of attribute names.

parameter
The parameter.

Discussion
If you implement this method you should also implement `accessibilityParameterizedAttributeNames` (page 3507).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSAccessibility.h`

accessibilityFocusedUIElement

Returns the deepest descendant of the accessibility hierarchy that has the focus.

```
- (id)accessibilityFocusedUIElement
```

Return Value
The deepest accessibility object in the accessibility hierarchy that has focus.

Discussion
You can assume that the search for the focus has already been narrowed down to the receiver. Override this method to do deeper searching by identifying which child element, if any, may have the focus. If a child element does not have the focus, either return `self` or, if available, invoke the superclass's implementation. The default `NSView` and `NSCell` implementations test whether the receiver is an ignored element and, if so, return the receiver's first unignored parent; otherwise they return `self`.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
Dicey
ImageMapExample
Sketch+Accessibility

TrackBall
ZipBrowser

Declared In
`NSAccessibility.h`

accessibilityHitTest:

Returns the deepest descendant of the accessibility hierarchy that contains the specified point.

```
- (id)accessibilityHitTest:(NSPoint)point
```

Parameters

point

The point being hit-tested, in lower-left relative screen coordinates.

Return Value
The deepest accessibility element in the accessibility hierarchy that contains the specified point.

Discussion
You can assume that the specified point has already been determined to lie within the receiver. Override this method to do deeper hit-testing by identifying which child element, if any, contains the point. `NSMatrix`, for example, identifies which of its cells contains the point and propagates the hit-test to it.

If the specified point is not contained within one of the receiver's children, either return `self` or, if available, invoke the superclass's implementation. The default `NSView` and `NSCell` implementations test whether the receiver is an ignored element and, if it is, return the receiver's first unignored parent; otherwise they return `self`.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
Dicey
ImageMap
ImageMapExample
Sketch+Accessibility

Declared In
`NSAccessibility.h`

accessibilityIndexOfChild:

Returns the index of the specified accessibility child in the parent.

```
- (NSUInteger)accessibilityIndexOfChild:(id)child
```

Parameters

child

The accessibility child of an object.

Return Value
The index of the accessibility child object in the parent. Returns `NSNotFound` if the child does not exist.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSAccessibility.h`

accessibilityIsAttributeSettable:

Returns a Boolean value that indicates whether the value for the specified attribute in the receiver can be set.

```
- (BOOL)accessibilityIsAttributeSettable:(NSString *)attribute
```

Parameters

attribute
The name of the attribute. See "Constants" (page 3509) for lists of attribute names.

Return Value
`YES` if the specified attribute can be set; otherwise, `NO`.

Discussion
User interface classes must implement this method. Subclasses should invoke the superclass's implementation, if it exists, if *attribute* is not implemented in the subclass.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSAccessibility.h`

accessibilityIsIgnored

Returns a Boolean value indicating whether the receiver should be ignored in the parent-child accessibility hierarchy.

```
- (BOOL)accessibilityIsIgnored
```

Return Value
`YES` if the receiver should be ignored; otherwise, `NO`.

Discussion
When asking for an object's children, ignored children should not be included; instead, the ignored children should be replaced by their own unignored children. The same applies when asking for an object's parent: an ignored parent should be skipped and the first unignored ancestor treated as the real parent. Likewise, when a hit-test or focus test is satisfied by an ignored element, the element's first unignored ancestor (or descendant in certain cases, such as single-celled controls) should be used instead.

Ignored elements allow the accessibility hierarchy to be a simplified version of the view and object ownership hierarchies. Intermediate objects can be bypassed and the real user interface objects accessed more quickly. For example, `NSControl` objects are ignored when they are single-celled; the visible parent-child relationship is between the control's parent (or a higher ancestor if the parent is ignored, too) and the control's cell.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
Dicey
ImageMap
ImageMapExample
Sketch+Accessibility
TrackBall

Declared In
`NSAccessibility.h`

accessibilityParameterizedAttributeNames

Returns a list of parameterized attribute names supported by the receiver.

```
- (NSArray *)accessibilityParameterizedAttributeNames
```

Return Value
An array of parameterized attributes in the receiver.

Discussion
If you implement this method you should also implement `accessibilityAttributeValue:forParameter:` (page 3504).

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
Sketch+Accessibility

Declared In
`NSAccessibility.h`

accessibilityPerformAction:

Performs the action associated with the specified action.

```
- (void)accessibilityPerformAction:(NSString *)action
```

Parameters
action
The action to perform.

Discussion
User interface classes must implement this method to handle all the actions returned from `accessibilityActionNames` (page 3501). Subclasses should invoke the superclass's implementation, if it exists, if *action* is not implemented in the subclass.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSAccessibility.h`

accessibilitySetOverrideValue:forAttribute:

Overrides the specified attribute in the receiver, or adds it if it does not exist, and sets its value to the specified value.

```
- (BOOL)accessibilitySetOverrideValue:(id)value forAttribute:(NSString *)attribute
```

Parameters

value

The attribute value to set.

attribute

The name of the attribute. See "Constants" (page 3509) for lists of attribute names.

Return Value

`YES` if the override was successful; otherwise, `NO`.

Discussion

This method is for changing the set of attributes on an instance, as an alternative to subclassing.

This method only works on objects whose class already implements the `NSAccessibility` protocol. If the specified attribute is already supported by the object, the value specified by this method wins.

If the specified attribute does not exist, it is created. This is done outside the `NSAccessibility` protocol, so `accessibilityAttributeNames` still returns the old list which does not contain the new attribute. Likewise, `accessibilityAttributeValue` does not return attributes created by the override process nor does it return their overridden values.

The values of overridden attributes are not settable by assistive applications.

If you need to undo the effect of using this method, call it again passing `nil` for the value.

Ensure that you invoke this method on the actual object that represents the user interface element. For example, in the case of `NSButton`, use the underlying `NSButtonCell`. The `NSButton` itself is ignored by accessibility.

This method works only on an object representing a single user interface element. So, for example, you cannot use it when a single object represents multiple user interface elements, as with `NSSegmentedCell`, which has only a single object but provides user interface elements for each segment.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

Dicey

Declared In

`NSAccessibility.h`

accessibilitySetValue:forAttribute:

Sets the value of the specified attribute in the receiver to the specified value.

```
- (void)accessibilitySetValue:(id)value forAttribute:(NSString *)attribute
```

Parameters

value

The attribute value to set.

attribute

The name of the attribute. See "Constants" (page 3509) for lists of attribute names.

Discussion

User interface classes must implement this method if any of its attributes are settable. Subclasses should invoke the superclass's implementation, if it exists, if *attribute* is not implemented in the subclass.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSAccessibility.h

Constants

Standard attributes

Standard attributes that can be adopted by any accessibility object.

```
NSString *const NSAccessibilityChildrenAttribute;
NSString *const NSAccessibilityContentsAttribute;
NSString *const NSAccessibilityDescriptionAttribute;
NSString *const NSAccessibilityEnabledAttribute;
NSString *const NSAccessibilityFocusedAttribute;
NSString *const NSAccessibilityHelpAttribute;
NSString *const NSAccessibilityMaxValueAttribute;
NSString *const NSAccessibilityMinValueAttribute;
NSString *const NSAccessibilityParentAttribute;
NSString *const NSAccessibilityPositionAttribute;
NSString *const NSAccessibilityRoleAttribute;
NSString *const NSAccessibilityRoleDescriptionAttribute;
NSString *const NSAccessibilitySelectedChildrenAttribute;
NSString *const NSAccessibilityShownMenuAttribute;
NSString *const NSAccessibilitySizeAttribute;
NSString *const NSAccessibilitySubroleAttribute;
NSString *const NSAccessibilityTitleAttribute;
NSString *const NSAccessibilityTopLevelUIElementAttribute;
NSString *const NSAccessibilityValueAttribute;
NSString *const NSAccessibilityValueDescriptionAttribute;
NSString *const NSAccessibilityVisibleChildrenAttribute;
NSString *const NSAccessibilityWindowAttribute;
```

Constants

NSAccessibilityChildrenAttribute

The element's child elements in the accessibility hierarchy (NSArray).

Available in Mac OS X v10.2 and later.

Declared in NSAccessibility.h.

`NSAccessibilityContentsAttribute`
Elements that represent the contents in the current element, such as the document view of a scroll view (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDescriptionAttribute`
The purpose of the element, not including the role (`NSString`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityEnabledAttribute`
A flag that indicates the enabled state of the element (`NSNumber`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFocusedAttribute`
A flag that indicates the presence of keyboard focus (`NSNumber`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityHelpAttribute`
The help text for the element (`NSString`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMaxValueAttribute`
The element's maximum value (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMinValueAttribute`
The element's minimum value (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityParentAttribute`
The element's parent element in the accessibility hierarchy (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityPositionAttribute`
The screen position of the element's lower-left corner in lower-left relative screen coordinates (`NSValue`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRoleAttribute`
The element's type, such as `NSAccessibilityRadioButtonRole` (`NSString`). See "Roles" (page 3533) for a list of available roles.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRoleDescriptionAttribute`
A localized, human-intelligible description of the element's role, such as "radio button" (`NSString`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySelectedChildrenAttribute`
The currently selected children of the element (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityShownMenuAttribute`
The menu currently being displayed (`id`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySizeAttribute`
The element's size (`NSValue`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySubroleAttribute`
The element's subrole, such as `NSAccessibilityTableRowSubrole` (`NSString`). See "Subroles" (page 3540) for a list of available subroles.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityTitleAttribute`
The title of the element, such as a button's visible text (`NSString`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityTopLevelUIElementAttribute`
The top-level element that contains this element (`id`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityValueAttribute`
The element's value (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityValueDescriptionAttribute`
The description of the element's value (`NSString`).

Available in Mac OS X v10.5 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityVisibleChildrenAttribute`
The element's child elements that are visible (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityWindowAttribute`
The window containing the current element (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Text-specific attributes

Attributes that are specific to text.

```
NSString *const NSAccessibilityInsertionPointLineNumberAttribute;
NSString *const NSAccessibilityNumberOfCharactersAttribute;
NSString *const NSAccessibilitySelectedTextAttribute;
NSString *const NSAccessibilitySelectedTextRangeAttribute;
NSString *const NSAccessibilitySelectedTextRangesAttribute;
NSString *const NSAccessibilitySharedCharacterRangeAttribute;
NSString *const NSAccessibilitySharedTextUIElementsAttribute;
NSString *const NSAccessibilityVisibleCharacterRangeAttribute;
```

Constants
`NSAccessibilityInsertionPointLineNumberAttribute`
The line number containing the insertion point (`NSNumber`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityNumberOfCharactersAttribute`
The number of characters in the text (`NSNumber`).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySelectedTextAttribute`
The currently selected text (`NSString`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySelectedTextRangeAttribute`
The range of selected text (`NSValue`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySelectedTextRangesAttribute`
An array of `NSValue` (`rangeValue`) ranges of selected text (`NSArray`).

Available in Mac OS X v10.5 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySharedCharacterRangeAttribute`
The (`rangeValue`) part of shared text in this view (`NSValue`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySharedTextUIElementsAttribute`
The elements with which the text of this element is shared (`NSArray`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityVisibleCharacterRangeAttribute`
The range of visible text (`NSValue`). Returns ranges for entire lines. For example, characters that are horizontally clipped will be reported in the visible range.

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Text-specific parameterized attributes

Parameterized attributes specific to text.

```
NSString *const NSAccessibilityAttributedStringForRangeParameterizedAttribute;
NSString *const NSAccessibilityBoundsForRangeParameterizedAttribute;
NSString *const NSAccessibilityLineForIndexParameterizedAttribute;
NSString *const NSAccessibilityRTFForRangeParameterizedAttribute;
NSString *const NSAccessibilityRangeForIndexParameterizedAttribute;
NSString *const NSAccessibilityRangeForLineParameterizedAttribute;
NSString *const NSAccessibilityRangeForPositionParameterizedAttribute;
NSString *const NSAccessibilityStringForRangeParameterizedAttribute;
NSString *const NSAccessibilityStyleRangeForIndexParameterizedAttribute;
```

Constants
`NSAccessibilityLineForIndexParameterizedAttribute`
The line number (`NSNumber`) of the specified character (`NSNumber`).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRangeForLineParameterizedAttribute`
The range of characters (`NSValue` containing an `NSRange`) corresponding to the specified line number (`NSNumber`).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityStringForRangeParameterizedAttribute`
The substring (`NSString`) specified by the range (`NSValue` containing an `NSRange`).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRangeForPositionParameterizedAttribute`
The range of characters (`NSValue` containing an `NSRange`) composing the glyph at the specified point (`NSValue` containing `NSPoint`).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRangeForIndexParameterizedAttribute`
: The full range of characters (`NSValue` containing an `NSRange`), including the specified character, which compose a single glyph (`NSNumber`).

  Available in Mac OS X v10.3 and later.

  Declared in `NSAccessibility.h`.

`NSAccessibilityBoundsForRangeParameterizedAttribute`
: The rectangle (`NSValue` containing an `NSRect`) enclosing the specified range of characters (`NSValue` containing an `NSRange`). If the range crosses a line boundary, the returned rectangle will fully enclose all the lines of characters.

  Available in Mac OS X v10.3 and later.

  Declared in `NSAccessibility.h`.

`NSAccessibilityRTFForRangeParameterizedAttribute`
: The RTF data (`NSData`) describing the specified range of characters (`NSValue` containing an `NSRange`).

  Available in Mac OS X v10.3 and later.

  Declared in `NSAccessibility.h`.

`NSAccessibilityStyleRangeForIndexParameterizedAttribute`
: The full range of characters (`NSValue` containing an `NSRange`), including the specified character (`NSNumber`), which have the same style.

  Available in Mac OS X v10.3 and later.

  Declared in `NSAccessibility.h`.

`NSAccessibilityAttributedStringForRangeParameterizedAttribute`
: Does not use attributes from Appkit/AttributedString.h (`NSAttributedString`).

  Available in Mac OS X v10.4 and later.

  Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Text attributed string attributes and constants

Attributes and key constants used with attributed strings.

```
NSString *const NSAccessibilityAttachmentTextAttribute;
NSString *const NSAccessibilityBackgroundColorTextAttribute;
NSString *const NSAccessibilityFontFamilyKey;
NSString *const NSAccessibilityFontNameKey;
NSString *const NSAccessibilityFontSizeKey;
NSString *const NSAccessibilityFontTextAttribute;
NSString *const NSAccessibilityForegroundColorTextAttribute;
NSString *const NSAccessibilityLinkTextAttribute;
NSString *const NSAccessibilityMisspelledTextAttribute;
NSString *const NSAccessibilityShadowTextAttribute;
NSString *const NSAccessibilityStrikethroughColorTextAttribute;
NSString *const NSAccessibilityStrikethroughTextAttribute;
NSString *const NSAccessibilitySuperscriptTextAttribute;
NSString *const NSAccessibilityUnderlineColorTextAttribute;
NSString *const NSAccessibilityUnderlineTextAttribute;
NSString *const NSAccessibilityVisibleNameKey;
```

Constants

`NSAccessibilityForegroundColorTextAttribute`
Text foreground color (`CGColorRef`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityBackgroundColorTextAttribute`
Text background color (`CGColorRef`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityUnderlineColorTextAttribute`
Text underline color (`CGColorRef`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityStrikethroughColorTextAttribute`
Text strikethrough color (`CGColorRef`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityUnderlineTextAttribute`
Text underline style (`NSNumber`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySuperscriptTextAttribute`
Text superscript style (`NSNumber`). Values > 0 are superscript; values < 0 are subscript.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityStrikethroughTextAttribute`
Text strikethrough (`NSNumber` as a Boolean value).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityShadowTextAttribute`
Text shadow (`NSNumber` as a Boolean value).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityAttachmentTextAttribute`
Text attachment (`id`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityLinkTextAttribute`
Text link (`id`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMisspelledTextAttribute`
Misspelled text (`NSNumber` as a Boolean value).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFontTextAttribute`
Font keys (`NSDictionary`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFontNameKey`
Required key for font name.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFontFamilyKey`
Optional key for font family.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityVisibleNameKey`
Optional key for font visibility.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFontSizeKey`
Required key for font size.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Window-specific attributes

Attributes specific to windows.

```
NSString *const NSAccessibilityCancelButtonAttribute;
NSString *const NSAccessibilityCloseButtonAttribute;
NSString *const NSAccessibilityDefaultButtonAttribute;
NSString *const NSAccessibilityGrowAreaAttribute;
NSString *const NSAccessibilityMainAttribute;
NSString *const NSAccessibilityMinimizeButtonAttribute;
NSString *const NSAccessibilityMinimizedAttribute;
NSString *const NSAccessibilityModalAttribute;
NSString *const NSAccessibilityProxyAttribute;
NSString *const NSAccessibilityToolbarButtonAttribute;
NSString *const NSAccessibilityZoomButtonAttribute;
```

Constants

`NSAccessibilityCloseButtonAttribute`

The element representing the close button (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityGrowAreaAttribute`

The element representing the grow area (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityModalAttribute`

A flag that indicates whether the window represented by this element is modal (`NSNumber`).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDefaultButtonAttribute`

The element that represents the default button (`id`).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityCancelButtonAttribute`

The element that represents the cancel button (`id`).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMainAttribute`

A flag that indicates whether the window is the main window (`NSNumber`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMinimizeButtonAttribute`

The element that represents the minimize button (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMinimizedAttribute`

A flag that indicates whether the window is minimized (`NSNumber`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityToolbarButtonAttribute`
The element that represents the toolbar button (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityProxyAttribute`
The element that represents the window's document proxy (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityZoomButtonAttribute`
The element that represents the zoom button (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Orientations

Values that indicate the orientation of elements, such as scroll bars and split views. One of these values is returned as the value for an object's `NSAccessibilityOrientationAttribute` (page 3532).

```
NSString *const NSAccessibilityHorizontalOrientationValue;
NSString *const NSAccessibilityVerticalOrientationValue;
NSString *const NSAccessibilityUnknownOrientationValue;
```

Constants
`NSAccessibilityHorizontalOrientationValue`
The element is oriented horizontally.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityVerticalOrientationValue`
The element is oriented vertically.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityUnknownOrientationValue`
The element has unknown orientation.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

Application-specific attributes

Attributes that are specific to the application object.

```
NSString *const NSAccessibilityClearButtonAttribute;
NSString *const NSAccessibilityColumnTitlesAttribute;
NSString *const NSAccessibilityFocusedUIElementAttribute;
NSString *const NSAccessibilityFocusedWindowAttribute;
NSString *const NSAccessibilityFrontmostAttribute;
NSString *const NSAccessibilityHiddenAttribute;
NSString *const NSAccessibilityMainWindowAttribute;
NSString *const NSAccessibilityMenuBarAttribute;
NSString *const NSAccessibilityOrientationAttribute;
NSString *const NSAccessibilitySearchButtonAttribute;
NSString *const NSAccessibilitySearchMenuAttribute;
NSString *const NSAccessibilityWindowsAttribute;
```

Constants

`NSAccessibilityFocusedUIElementAttribute`

The element with the current focus (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFocusedWindowAttribute`

The application's window that has current focus (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFrontmostAttribute`

A flag that indicates whether the application is frontmost (`NSNumber`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityHiddenAttribute`

A flag that indicates whether the application is hidden (`NSNumber`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMainWindowAttribute`

The application's main window (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMenuBarAttribute`

The application's menu bar (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityWindowsAttribute`

The application's windows (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

Declared In

`NSAccessibility.h`

Grid view attributes

Attributes that are used with grid views, such as thumbnails and media browsers that present a grid of items. The children of a grid are ordered.

```
NSString *const NSAccessibilityColumnCountAttribute;
NSString *const NSAccessibilityOrderedByRowAttribute;
NSString *const NSAccessibilityRowCountAttribute;
```

Constants

`NSAccessibilityColumnCountAttribute`

The number of columns in the grid (`NSNumber` as `intValue`).

Available in Mac OS X v10.5 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityOrderedByRowAttribute`

A flag that indicates whether the grid is ordered row major (`YES`), or column major (`NO`) (`NSNumber` as `boolValue`).

Available in Mac OS X v10.5 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRowCountAttribute`

The number of rows in the grid (`NSNumber` as `intValue`).

Available in Mac OS X v10.5 and later.

Declared in `NSAccessibility.h`.

Declared In

`NSAccessibility.h`

Table view and outline view attributes

Attributes that are specific to tables and outlines.

```
NSString *const NSAccessibilityColumnHeaderUIElementsAttribute;
NSString *const NSAccessibilityColumnsAttribute;
NSString *const NSAccessibilityRowHeaderUIElementsAttribute;
NSString *const NSAccessibilityRowsAttribute;
NSString *const NSAccessibilitySelectedColumnsAttribute;
NSString *const NSAccessibilitySelectedRowsAttribute;
NSString *const NSAccessibilitySortDirectionAttribute;
NSString *const NSAccessibilityVisibleColumnsAttribute;
NSString *const NSAccessibilityVisibleRowsAttribute;
```

Constants

`NSAccessibilityColumnHeaderUIElementsAttribute`

The table's column headers (`NSArray`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityColumnsAttribute`

The table's columns (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRowHeaderUIElementsAttribute`
The table's row headers (`NSArray`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRowsAttribute`
The table's rows (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySelectedColumnsAttribute`
The table's selected columns (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySelectedRowsAttribute`
The table's selected rows (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySortDirectionAttribute`
The column's sort direction (`NSString`). See "`Column sort direction`" (page 3523) for possible values.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityVisibleColumnsAttribute`
The table's visible columns (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityVisibleRowsAttribute`
The table's visible rows (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Outline view attributes

Attributes that are used in outline views.

```
NSString *const NSAccessibilityDisclosedByRowAttribute;
NSString *const NSAccessibilityDisclosedRowsAttribute;
NSString *const NSAccessibilityDisclosingAttribute;
NSString *const NSAccessibilityDisclosureLevelAttribute;
```

Constants

`NSAccessibilityDisclosedByRowAttribute`

The row disclosing this row (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDisclosedRowsAttribute`

The rows disclosed by this row (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDisclosingAttribute`

A flag that indicates whether a row is disclosing other rows (`NSNumber`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDisclosureLevelAttribute`

The indentation level of this row (`NSNumber`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

Declared In

`NSAccessibility.h`

Cell-based table attributes

Attributes that are specific to cell-based tables.

```
NSString *const NSAccessibilitySelectedCellsAttribute;
NSString *const NSAccessibilityVisibleCellsAttribute;
```

Constants

`NSAccessibilitySelectedCellsAttribute`

The table's selected cells (`NSArray`). This attribute is required for cell-based tables.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityVisibleCellsAttribute`

The table's visible cells (`NSArray`). This attribute is required for cell-based tables.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

Cell-based table parameterized attributes

Parameterized attributes specific to cell-based tables.

```
NSString *const NSAccessibilityCellForColumnAndRowParameterizedAttribute;
```

Constants

`NSAccessibilityCellForColumnAndRowParameterizedAttribute`

The cell at the specified row and column. The parameter is an `NSArray` that contains two `NSNumber` objects: the first number specifies the column index and the second number specifies the row index. This attribute is required for cell-based tables.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

Cell attributes

Attributes that are specific to individual table cells.

```
NSString *const NSAccessibilityRowIndexRangeAttribute;
NSString *const NSAccessibilityColumnIndexRangeAttribute;
```

Constants

`NSAccessibilityRowIndexRangeAttribute`

The row index range of the cell (an `NSValue` that contains the row's starting index and index span in the table).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityColumnIndexRangeAttribute`

The column index range of the cell (an `NSValue` that contains the row's starting index and index span in the table).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

Column sort direction

Values that indicate the sort direction of a column (used with `NSAccessibilitySortDirectionAttribute` (page 3521)).

```
NSString *const NSAccessibilityAscendingSortDirectionValue;
NSString *const NSAccessibilityDescendingSortDirectionValue;
NSString *const NSAccessibilityUnknownSortDirectionValue;
```

Constants

`NSAccessibilityAscendingSortDirectionValue`

The column is sorted in ascending values.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDescendingSortDirectionValue`

The column is sorted in descending values.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityUnknownSortDirectionValue`
The sort direction is unknown.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Layout area attributes

Attributes that are specific to layout areas.

```
NSString *const NSAccessibilityHorizontalUnitsAttribute;
NSString *const NSAccessibilityVerticalUnitsAttribute;
NSString *const NSAccessibilityHorizontalUnitDescriptionAttribute;
NSString *const NSAccessibilityVerticalUnitDescriptionAttribute;
```

Constants
`NSAccessibilityHorizontalUnitsAttribute`
The units that the layout view uses for horizontal values (`NSString`). See "Ruler unit attributes" (page 3529) for possible values.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityVerticalUnitsAttribute`
The units that the layout view uses for vertical values (`NSString`). See "Ruler unit attributes" (page 3529) for possible values.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityHorizontalUnitDescriptionAttribute`
The description of the layout view's horizontal units (`NSString`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityVerticalUnitDescriptionAttribute`
The description of the layout view's vertical units (`NSString`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

Layout area parameterized attributes

Parameterized attributes that are specific to layout areas.

```
NSString *const NSAccessibilityLayoutPointForScreenPointParameterizedAttribute;
NSString *const NSAccessibilityLayoutSizeForScreenSizeParameterizedAttribute;
NSString *const NSAccessibilityScreenPointForLayoutPointParameterizedAttribute;
NSString *const NSAccessibilityScreenSizeForLayoutSizeParameterizedAttribute;
```

Constants

`NSAccessibilityLayoutPointForScreenPointParameterizedAttribute`

The point in the layout area (`NSValue`) corresponding to the specified point on the screen (`NSValue`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityLayoutSizeForScreenSizeParameterizedAttribute`

The size of the layout area (`NSValue`) corresponding to the specified screen size (`NSValue`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityScreenPointForLayoutPointParameterizedAttribute`

The screen point (`NSValue`) corresponding to the specified point in the layout area (`NSValue`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityScreenSizeForLayoutSizeParameterizedAttribute`

The size of the screen (`NSValue`) corresponding to the specified size of the layout area (`NSValue`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

Layout item attributes

Attributes that are specific to the items in a layout area.

```
NSString *const NSAccessibilityHandlesAttribute;
```

Constants

`NSAccessibilityHandlesAttribute`

The drag handles of the item (`NSArray`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

Slider attributes

Attributes that are specific to sliders.

```
NSString *const NSAccessibilityAllowedValuesAttribute;
NSString *const NSAccessibilityLabelUIElementsAttribute;
NSString *const NSAccessibilityLabelValueAttribute;
```

Constants

`NSAccessibilityAllowedValuesAttribute`

The allowed values in the slider (`NSArray`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityLabelUIElementsAttribute`

The elements that represent the slider's labels (`NSArray`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityLabelValueAttribute`

The value of the label represented by this element (`NSNumber`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

Declared In

`NSAccessibility.h`

Screen matte attributes

Attributes that are specific to screen mattes.

```
NSString *const NSAccessibilityMatteContentUIElementAttribute;
NSString *const NSAccessibilityMatteHoleAttribute;
```

Constants

`NSAccessibilityMatteHoleAttribute`

The bounds of the matte hole, in screen coordinates (`NSValue` containing an `NSRect`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMatteContentUIElementAttribute`

The element that is clipped by the matte (`id`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

Declared In

`NSAccessibility.h`

Ruler view attributes

Attributes that are specific to ruler views.

```
NSString *const NSAccessibilityMarkerGroupUIElementAttribute;
NSString *const NSAccessibilityMarkerTypeAttribute;
NSString *const NSAccessibilityMarkerTypeDescriptionAttribute;
NSString *const NSAccessibilityMarkerUIElementsAttribute;
NSString *const NSAccessibilityMarkerValuesAttribute;
NSString *const NSAccessibilityUnitDescriptionAttribute;
NSString *const NSAccessibilityUnitsAttribute;
```

Constants

`NSAccessibilityMarkerGroupUIElementAttribute`

Marker group user interface element (`id`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMarkerTypeAttribute`

The type of the marker (`NSString`). See "Ruler marker type values" (page 3527) for possible values.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMarkerTypeDescriptionAttribute`

The description of the marker type (`NSString`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMarkerUIElementsAttribute`

Array of marker user interface elements (`NSArray`)

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMarkerValuesAttribute`

The marker values (`NSArray` of `NSNumber`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityUnitDescriptionAttribute`

The description of ruler units (`NSString`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityUnitsAttribute`

The ruler units (`NSString`). See "Ruler unit attributes" (page 3529) for possible values.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

Declared In

`NSAccessibility.h`

Ruler marker type values

Values that indicate the marker type of an element.

```
NSString *const NSAccessibilityCenterTabStopMarkerTypeValue;
NSString *const NSAccessibilityDecimalTabStopMarkerTypeValue;
NSString *const NSAccessibilityFirstLineIndentMarkerTypeValue;
NSString *const NSAccessibilityHeadIndentMarkerTypeValue;
NSString *const NSAccessibilityLeftTabStopMarkerTypeValue;
NSString *const NSAccessibilityRightTabStopMarkerTypeValue;
NSString *const NSAccessibilityTailIndentMarkerTypeValue;
NSString *const NSAccessibilityUnknownMarkerTypeValue;
```

Constants

`NSAccessibilityLeftTabStopMarkerTypeValue`

Left tab stop.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRightTabStopMarkerTypeValue`

Right tab stop.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityCenterTabStopMarkerTypeValue`

Center tab stop.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDecimalTabStopMarkerTypeValue`

Decimal tab stop.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityHeadIndentMarkerTypeValue`

Head indent marker.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityTailIndentMarkerTypeValue`

Tail indent marker.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFirstLineIndentMarkerTypeValue`

First line indent marker.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityUnknownMarkerTypeValue`

Unknown marker type.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

Declared In

`NSAccessibility.h`

Measurement unit attributes

Values that indicate the unit values of a ruler or layout area (used with `NSAccessibilityUnitsAttribute` (page 3527)).

```
NSString *const NSAccessibilityCentimetersUnitValue;
NSString *const NSAccessibilityInchesUnitValue;
NSString *const NSAccessibilityPicasUnitValue;
NSString *const NSAccessibilityPointsUnitValue;
NSString *const NSAccessibilityUnknownUnitValue;
```

Constants

`NSAccessibilityInchesUnitValue`
The units are inches.
Available in Mac OS X v10.4 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityCentimetersUnitValue`
The units are centimeters.
Available in Mac OS X v10.4 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityPointsUnitValue`
The units are points.
Available in Mac OS X v10.4 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityPicasUnitValue`
The units are picas.
Available in Mac OS X v10.4 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityUnknownUnitValue`
The units are unknown.
Available in Mac OS X v10.4 and later.
Declared in `NSAccessibility.h`.

Declared In

`NSAccessibility.h`

Linkage elements

Constants that specify links between accessibility elements.

```
NSString *const NSAccessibilityLinkedUIElementsAttribute;
NSString *const NSAccessibilityServesAsTitleForUIElementsAttribute;
NSString *const NSAccessibilityTitleUIElementAttribute;
```

Constants

`NSAccessibilityLinkedUIElementsAttribute`

The elements with which this element is related (`NSArray`). For example, the contents of a list item that are displayed in another pane or window.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityTitleUIElementAttribute`

An element that represents another element's static text title (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityServesAsTitleForUIElementsAttribute`

The elements for which this element serves as the title (`NSArray`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

Declared In

`NSAccessibility.h`

Miscellaneous attributes

Miscellaneous attributes that can apply to various types of elements.

```
NSString *const NSAccessibilityDecrementButtonAttribute;
NSString *const NSAccessibilityDocumentAttribute;
NSString *const NSAccessibilityEditedAttribute;
NSString *const NSAccessibilityExpandedAttribute;
NSString *const NSAccessibilityFilenameAttribute;
NSString *const NSAccessibilityHeaderAttribute;
NSString *const NSAccessibilityHorizontalScrollBarAttribute;
NSString *const NSAccessibilityIncrementButtonAttribute;
NSString *const NSAccessibilityIndexAttribute;
NSString *const NSAccessibilityNextContentsAttribute;
NSString *const NSAccessibilityOverflowButtonAttribute;
NSString *const NSAccessibilityPreviousContentsAttribute;
NSString *const NSAccessibilitySelectedAttribute;
NSString *const NSAccessibilitySplittersAttribute;
NSString *const NSAccessibilityTabsAttribute;
NSString *const NSAccessibilityURLAttribute;
NSString *const NSAccessibilityVerticalScrollBarAttribute;
NSString *const NSAccessibilityWarningValueAttribute;
NSString *const NSAccessibilityCriticalValueAttribute;
NSString *const NSAccessibilityPlaceholderValueAttribute;
```

Constants

`NSAccessibilityClearButtonAttribute`

The element that represents the clear button in a search field (`id`).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityColumnTitlesAttribute`
The elements that represent column titles (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDecrementButtonAttribute`
The element that represents a stepper's decrement button (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDocumentAttribute`
The URL for the file represented by the element (`NSString`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityEditedAttribute`
A flag that indicates whether the element has been modified (`NSNumber`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityExpandedAttribute`
A flag that indicates whether the element is expanded (`NSNumber`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFilenameAttribute`
The filename associated with the element (`NSString`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityHeaderAttribute`
The element that represents a table view's header (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityHorizontalScrollBarAttribute`
The element that represents a scroll view's horizontal scroll bar (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityIncrementButtonAttribute`
The element that represents a stepper's increment button (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityIndexAttribute`
The index of the row or column represented by the element (`NSValue`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityNextContentsAttribute`
The contents following the current divider element, such as a subview adjacent to a split view's splitter element (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityOrientationAttribute`
The element's orientation, which can have the value `NSAccessibilityHorizontalOrientationValue` (page 3518) or `NSAccessibilityVerticalOrientationValue` (page 3518).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityOverflowButtonAttribute`
The element that represents a toolbar's overflow button (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityPreviousContentsAttribute`
The contents preceding the current divider element, such as a subview adjacent to a split view's splitter bar element (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySearchButtonAttribute`
The element that represents the search button in a search field (`id`).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySearchMenuAttribute`
The element that represents the menu in a search field (`id`).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySelectedAttribute`
A flag that indicates whether the element is selected (`NSNumber`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySplittersAttribute`
The views and splitter bar in a split view (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityTabsAttribute`
The tab elements in a tab view (`NSArray`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityURLAttribute`
The URL associated with the element (`NSURL`).

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityVerticalScrollBarAttribute`
The element that represents the vertical scroll bar in a scroll view (`id`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityWarningValueAttribute`
The warning value in a level indicator (typically, `NSNumber`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityCriticalValueAttribute`
The critical value in a level indicator (typically, `NSNumber`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityPlaceholderValueAttribute`
The placeholder value for a control, such as a text field (`NSString`).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Roles

Standard roles that identify the type of object an accessibility object represents. One of these values is returned as the value for an object's `NSAccessibilityRoleAttribute` (page 3510).

```
NSString *const NSAccessibilityApplicationRole;
NSString *const NSAccessibilityBrowserRole;
NSString *const NSAccessibilityBusyIndicatorRole;
NSString *const NSAccessibilityButtonRole;
NSString *const NSAccessibilityCellRole;
NSString *const NSAccessibilityCheckBoxRole;
NSString *const NSAccessibilityColorWellRole;
NSString *const NSAccessibilityColumnRole;
NSString *const NSAccessibilityComboBoxRole;
NSString *const NSAccessibilityDisclosureTriangleRole;
NSString *const NSAccessibilityDrawerRole;
NSString *const NSAccessibilityGridRole;
NSString *const NSAccessibilityGroupRole;
NSString *const NSAccessibilityGrowAreaRole;
NSString *const NSAccessibilityHandleRole;
NSString *const NSAccessibilityHelpTagRole;
NSString *const NSAccessibilityImageRole;
NSString *const NSAccessibilityIncrementorRole;
NSString *const NSAccessibilityLayoutAreaRole;
NSString *const NSAccessibilityLayoutItemRole;
NSString *const NSAccessibilityLinkRole;
NSString *const NSAccessibilityListRole;
NSString *const NSAccessibilityMatteRole;
NSString *const NSAccessibilityMenuBarRole;
NSString *const NSAccessibilityMenuButtonRole;
NSString *const NSAccessibilityMenuItemRole;
NSString *const NSAccessibilityMenuRole;
NSString *const NSAccessibilityOutlineRole;
NSString *const NSAccessibilityPopUpButtonRole;
NSString *const NSAccessibilityProgressIndicatorRole;
NSString *const NSAccessibilityRadioButtonRole;
NSString *const NSAccessibilityRadioGroupRole;
NSString *const NSAccessibilityRelevanceIndicatorRole;
NSString *const NSAccessibilityRowRole;
NSString *const NSAccessibilityRulerMarkerRole;
NSString *const NSAccessibilityRulerRole;
NSString *const NSAccessibilityScrollAreaRole;
NSString *const NSAccessibilityScrollBarRole;
NSString *const NSAccessibilitySheetRole;
NSString *const NSAccessibilitySliderRole;
NSString *const NSAccessibilitySortButtonRole;
NSString *const NSAccessibilitySplitGroupRole;
NSString *const NSAccessibilitySplitterRole;
NSString *const NSAccessibilityStaticTextRole;
NSString *const NSAccessibilitySystemWideRole;
NSString *const NSAccessibilityTabGroupRole;
NSString *const NSAccessibilityTableRole;
NSString *const NSAccessibilityTextAreaRole;
NSString *const NSAccessibilityTextFieldRole;
NSString *const NSAccessibilityToolbarRole;
NSString *const NSAccessibilityUnknownRole;
NSString *const NSAccessibilityValueIndicatorRole;
NSString *const NSAccessibilityWindowRole;
```

Constants

`NSAccessibilityApplicationRole`
Application.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityBrowserRole`
Browser.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityBusyIndicatorRole`
Busy indicator.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityButtonRole`
Button.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityCellRole`
Cell in a table or matrix.
Available in Mac OS X v10.6 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityCheckBoxRole`
Checkbox.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityColorWellRole`
Color well.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityColumnRole`
Column.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityComboBoxRole`
Combo box.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityDisclosureTriangleRole`
Disclosure triangle.
Available in Mac OS X v10.5 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityDrawerRole`
Drawer.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityGridRole`
Grid.

Available in Mac OS X v10.5 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityGroupRole`
Group.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityGrowAreaRole`
A window's grow (resize) area.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityHandleRole`
Drag handle.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityHelpTagRole`
Help tag.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityImageRole`
Image.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityIncrementorRole`
Stepper.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityLayoutAreaRole`
Layout area (a view, such as a graphic view, that contains visual elements that may not have any accessibility representation).

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityLayoutItemRole`
An item in a layout area.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityLinkRole`
Link.
Available in Mac OS X v10.4 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityListRole`
List.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityMatteRole`
Matte.
Available in Mac OS X v10.4 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityMenuBarRole`
Menu bar.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityMenuButtonRole`
Menu button.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityMenuItemRole`
Menu item.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityMenuRole`
Menu.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityOutlineRole`
Outline.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityPopUpButtonRole`
Pop-up button.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityProgressIndicatorRole`
Progress indicator.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityRadioButtonRole`
Radio button.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityRadioGroupRole`
Radio button group.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityRelevanceIndicatorRole`
Relevance indicator.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityRowRole`
Row.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityRulerRole`
Ruler.
Available in Mac OS X v10.4 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityRulerMarkerRole`
Ruler marker.
Available in Mac OS X v10.4 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityScrollAreaRole`
Scroll view.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityScrollBarRole`
Scroll bar.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilitySheetRole`
Sheet.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilitySliderRole`
Slider.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilitySortButtonRole`
Sort button.
Available in Mac OS X v10.4 and later.
Deprecated in Mac OS X v10.6.
Declared in `NSAccessibility.h`.

`NSAccessibilitySplitGroupRole`
Split view.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilitySplitterRole`
Splitter bar of a split view.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityStaticTextRole`
Uneditable text.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilitySystemWideRole`
The system-wide accessibility object.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityTabGroupRole`
Tab group.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityTableRole`
Table.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityTextAreaRole`
Text view.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityTextFieldRole`
Text field.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityToolbarRole`
Toolbar.
Available in Mac OS X v10.2 and later.
Declared in `NSAccessibility.h`.

`NSAccessibilityUnknownRole`
Unknown object type.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityValueIndicatorRole`
Value indicator.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityWindowRole`
Window.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Subroles

Subroles that identify a specialized type of object an accessibility object represents. One of these values is returned as the value for an object's `NSAccessibilitySubroleAttribute` (page 3511).

```
NSString *const NSAccessibilityCloseButtonSubrole;
NSString *const NSAccessibilityDecrementArrowSubrole;
NSString *const NSAccessibilityDecrementPageSubrole;
NSString *const NSAccessibilityDialogSubrole;
NSString *const NSAccessibilityFloatingWindowSubrole;
NSString *const NSAccessibilityIncrementArrowSubrole;
NSString *const NSAccessibilityIncrementPageSubrole;
NSString *const NSAccessibilityMinimizeButtonSubrole;
NSString *const NSAccessibilityOutlineRowSubrole;
NSString *const NSAccessibilitySearchFieldSubrole;
NSString *const NSAccessibilitySecureTextFieldSubrole;
NSString *const NSAccessibilityStandardWindowSubrole;
NSString *const NSAccessibilitySystemDialogSubrole;
NSString *const NSAccessibilitySystemFloatingWindowSubrole;
NSString *const NSAccessibilityTableRowSubrole;
NSString *const NSAccessibilityTextAttachmentSubrole;
NSString *const NSAccessibilityTextLinkSubrole;
NSString *const NSAccessibilityTimelineSubrole;
NSString *const NSAccessibilityToolbarButtonSubrole;
NSString *const NSAccessibilityUnknownSubrole;
NSString *const NSAccessibilityZoomButtonSubrole;
NSString *const NSAccessibilitySortButtonSubrole;
NSString *const NSAccessibilityRatingIndicatorSubrole;
NSString *const NSAccessibilityContentListSubrole;
NSString *const NSAccessibilityDefinitionListSubrole;
```

Constants
`NSAccessibilityCloseButtonSubrole`
A window's close button.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDecrementArrowSubrole`
Decrement arrow (the down arrow in a scroll bar).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDecrementPageSubrole`
Decrement page (the decrement area in the scroll track of a scroll bar).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDialogSubrole`
Dialog.

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFloatingWindowSubrole`
Floating window.

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityIncrementArrowSubrole`
Increment arrow (the up arrow in a scroll bar).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityIncrementPageSubrole`
Increment page (the increment area in the scroll track of a scroll bar).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMinimizeButtonSubrole`
A window's minimize button.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityOutlineRowSubrole`
Outline row.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySearchFieldSubrole`
Search field.

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySecureTextFieldSubrole`
Secure text field.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityStandardWindowSubrole`
A standard window.

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySystemDialogSubrole`
System dialog (a system-generated dialog that floats on the top layer, regardless of which application is frontmost).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySystemFloatingWindowSubrole`
System floating window (a system-generated panel).

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityTableRowSubrole`
Table row.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityTextAttachmentSubrole`
Text attachment.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityTextLinkSubrole`
Text link.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityTimelineSubrole`
Timeline.

Available in Mac OS X v10.5 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityToolbarButtonSubrole`
A window's toolbar button.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityUnknownSubrole`
Unknown subrole.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityZoomButtonSubrole`
A window's zoom button.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySortButtonSubrole`
Sort button in a table or outline view.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRatingIndicatorSubrole`
Rating indicator.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityContentListSubrole`
Content that is organized in a list, but is not in a list control or table view.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDefinitionListSubrole`
A content list in a webpage.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Action values

Standard actions that accessibility objects can perform.

```
NSString *const NSAccessibilityCancelAction;
NSString *const NSAccessibilityConfirmAction;
NSString *const NSAccessibilityDecrementAction;
NSString *const NSAccessibilityDeleteAction;
NSString *const NSAccessibilityIncrementAction;
NSString *const NSAccessibilityPickAction;
NSString *const NSAccessibilityPressAction;
NSString *const NSAccessibilityRaiseAction;
NSString *const NSAccessibilityShowMenuAction;
```

Constants
`NSAccessibilityConfirmAction`
Simulates pressing Return in the object, such as a text field.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDecrementAction`
Decrements the value of the object.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityDeleteAction`
Deletes the value of the object.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityIncrementAction`
Increments the value of the object.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityPickAction`
Selects the object, such as a menu item.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityPressAction`
Simulates clicking an object, such as a button.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityCancelAction`
Cancels the operation.

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRaiseAction`
Simulates bringing a window forward by clicking on its title bar.

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityShowMenuAction`
Simulates showing a menu by clicking on it.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Focus-change notifications

Notifications that are sent to observing assistive applications when focus-change events occur. The notifications are sent using the `NSAccessibilityPostNotification` (page 3956) function instead of an `NSNotificationCenter` object.

```
NSString *const NSAccessibilityMainWindowChangedNotification;
NSString *const NSAccessibilityFocusedWindowChangedNotification;
NSString *const NSAccessibilityFocusedUIElementChangedNotification;
```

Constants
`NSAccessibilityMainWindowChangedNotification`
Posted after the main window has changed.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFocusedWindowChangedNotification`
Posted after the key window has changed.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityFocusedUIElementChangedNotification`
Posted after the element has gained focus.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Window-change notifications

Notifications that are sent to observing assistive applications when window-change events occur. The notifications are sent using the `NSAccessibilityPostNotification` (page 3956) function instead of an `NSNotificationCenter` object.

```
NSString *const NSAccessibilityWindowCreatedNotification;
NSString *const NSAccessibilityWindowDeminiaturizedNotification;
NSString *const NSAccessibilityWindowMiniaturizedNotification;
NSString *const NSAccessibilityWindowMovedNotification;
NSString *const NSAccessibilityWindowResizedNotification;
```

Constants
`NSAccessibilityWindowCreatedNotification`
Posted after a new window has appeared.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityWindowDeminiaturizedNotification`
Posted after the window has been restored to full size from the Dock.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityWindowMiniaturizedNotification`
Posted after the window has been put in the Dock.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityWindowMovedNotification`
Posted after the window has moved.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityWindowResizedNotification`
Posted after the window has changed size.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Application notifications

Notifications that are sent to observing assistive applications when application events occur. The notifications are sent using the `NSAccessibilityPostNotification` (page 3956) function instead of an `NSNotificationCenter` object.

```
NSString *const NSAccessibilityApplicationActivatedNotification;
NSString *const NSAccessibilityApplicationDeactivatedNotification;
NSString *const NSAccessibilityApplicationHiddenNotification;
NSString *const NSAccessibilityApplicationShownNotification;
```

Constants

`NSAccessibilityApplicationActivatedNotification`

Posted after the application has activated.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityApplicationDeactivatedNotification`

Posted after the application has deactivated.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityApplicationHiddenNotification`

Posted after the application has been hidden.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityApplicationShownNotification`

Posted after the application has been shown.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

Declared In

`NSAccessibility.h`

Drawer and sheet notifications

Notifications that are sent to observing assistive applications when drawer and sheet events occur. The notifications are sent using the `NSAccessibilityPostNotification` (page 3956) function instead of an `NSNotificationCenter` object.

```
NSString *const NSAccessibilityDrawerCreatedNotification;
NSString *const NSAccessibilitySheetCreatedNotification;
```

Constants

`NSAccessibilityDrawerCreatedNotification`

Posted after a drawer has appeared.

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySheetCreatedNotification`
Posted after a sheet has appeared.

Available in Mac OS X v10.3 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

Element notifications

Notifications that are sent to observing assistive applications when certain element-related events occur. Note that these notifications are not sent from every element. The notifications are sent using the `NSAccessibilityPostNotification` (page 3956) function instead of an `NSNotificationCenter` object.

```
NSString *const NSAccessibilityCreatedNotification;
NSString *const NSAccessibilityMovedNotification;
NSString *const NSAccessibilityResizedNotification;
NSString *const NSAccessibilityTitleChangedNotification;
NSString *const NSAccessibilityUIElementDestroyedNotification;
NSString *const NSAccessibilityValueChangedNotification;
```

Constants
`NSAccessibilityCreatedNotification`
Posted after the element has been created.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityMovedNotification`
Posted after the element has been moved.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityResizedNotification`
Posted after the element has been resized.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityTitleChangedNotification`
Posted after the title has changed.

Available in Mac OS X v10.4 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityUIElementDestroyedNotification`
Posted after the element has been destroyed.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityValueChangedNotification`
Posted after the element's value has changed.

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

Declared In
NSAccessibility.h

Miscellaneous notifications

Notifications that are sent to observing assistive applications when certain events occur. The notifications are sent using the NSAccessibilityPostNotification (page 3956) function instead of an NSNotificationCenter object.

```
NSString *const NSAccessibilityHelpTagCreatedNotification;
NSString *const NSAccessibilityRowCountChangedNotification;
NSString *const NSAccessibilitySelectedChildrenChangedNotification;
NSString *const NSAccessibilitySelectedColumnsChangedNotification;
NSString *const NSAccessibilitySelectedRowsChangedNotification;
NSString *const NSAccessibilitySelectedTextChangedNotification;
NSString *const NSAccessibilityRowExpandedNotification;
NSString *const NSAccessibilityRowCollapsedNotification;
NSString *const NSAccessibilitySelectedCellsChangedNotification;
NSString *const NSAccessibilityUnitsChangedNotification;
NSString *const NSAccessibilitySelectedChildrenMovedNotification;
```

Constants

NSAccessibilityHelpTagCreatedNotification
Posted after a help tag has appeared.
Available in Mac OS X v10.4 and later.
Declared in NSAccessibility.h.

NSAccessibilityRowCountChangedNotification
Posted after a row has been added or deleted.
Available in Mac OS X v10.4 and later.
Declared in NSAccessibility.h.

NSAccessibilitySelectedChildrenChangedNotification
Posted after selected child elements have changed.
Available in Mac OS X v10.4 and later.
Declared in NSAccessibility.h.

NSAccessibilitySelectedColumnsChangedNotification
Posted after selected columns have changed.
Available in Mac OS X v10.4 and later.
Declared in NSAccessibility.h.

NSAccessibilitySelectedRowsChangedNotification
Posted after selected rows have changed.
Available in Mac OS X v10.4 and later.
Declared in NSAccessibility.h.

NSAccessibilitySelectedTextChangedNotification
Posted after selected text has changed.
Available in Mac OS X v10.4 and later.
Declared in NSAccessibility.h.

`NSAccessibilityRowExpandedNotification`
Posted after the row has expanded.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityRowCollapsedNotification`
Posted after the row has collapsed.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySelectedCellsChangedNotification`
Posted after selected cells in a cell-based table have changed.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilityUnitsChangedNotification`
Posted after the units in a layout area have changed.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

`NSAccessibilitySelectedChildrenMovedNotification`
Posted after selected items in a layout area have moved.

Available in Mac OS X v10.6 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

UserInfo key for error codes in accessibility exceptions

This is a key used by the `userInfo` dictionary of an `NSAccessibilityException`.

```
NSString *const NSAccessibilityErrorCodeExceptionInfo;
```

Constants
`NSAccessibilityErrorCodeExceptionInfo`
Integer error code used for debugging (as an `NSNumber`).

Available in Mac OS X v10.2 and later.

Declared in `NSAccessibility.h`.

Declared In
`NSAccessibility.h`

NSAlertDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSAlert.h
Companion guides	Dialogs and Special Panels Sheet Programming Topics

Overview

The `NSAlertDelegate` protocol defines the optional methods implemented by delegates of `NSAlert` objects.

Tasks

Displaying Help

`- alertShowHelp:` (page 3551)
Sent to the delegate when the user clicks the alert's help button. The delegate causes help to be displayed for an alert, directly or indirectly.

Instance Methods

alertShowHelp:

Sent to the delegate when the user clicks the alert's help button. The delegate causes help to be displayed for an alert, directly or indirectly.

```
- (BOOL)alertShowHelp:(NSAlert *)alert
```

Return Value
`YES` when the delegate displayed help directly, `NO` otherwise. When `NO` and the alert has a help anchor (`setHelpAnchor:` (page 90)), the application's help manager displays help using the help anchor.

Discussion
The delegate implements this method only to override the help-anchor lookup behavior.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- setShowsHelp:` (page 92) (NSAlert)

Declared In
`NSAlert.h`

NSAnimatablePropertyContainer Protocol Reference

Adopted by	NSWindow NSView
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSAnimation.h

Overview

The NSAnimatablePropertyContainer protocol defines a way to add animation to an existing class with a minimum of API impact. It returns a proxy object for the receiver that can be used to initiate implied animation of property changes. An object's animator proxy should be treated as if it was the object itself, and may be passed to any code that accepts the object as a parameter. Sending of key-value-coding compliant "set" messages to the proxy will trigger animation for automatically animated properties of its target object.

An object's automatically animated properties are those for which `NSAnimatablePropertyContainer` (page 3553) finds and returns an CAAnimation instead of nil, often because `animator` (page 3556) specifies a default animation for the key.

It's perfectly valid to set a new value for a property for which an animation that is currently in progress; this simply sets a new target value for that property, with animation to the new target proceeding from whatever current value the property has reached. An in-flight property animation can be stopped by setting a new value for the property bracketed by an `NSAnimationContext` with 0.0 as the duration.

Tasks

Getting the Animator Proxy

- `animator` (page 3556) *required method*

  Returns a proxy object for the receiver that can be used to initiate implied animation for property changes. (required)

Managing Animations for Properties

- `animations` (page 3555) *required method*
  Returns the optional dictionary that maps event trigger keys to animation objects. (required)
- `setAnimations:` (page 3557) *required method*
  Sets the option dictionary that maps event trigger keys to animation objects. (required)
- `animationForKey:` (page 3555) *required method*
  Returns the animation that should be performed for the specified key. (required)
+ `defaultAnimationForKey:` (page 3554) *required method*
  Returns the default animation that should be performed for the specified key. (required)

Class Methods

defaultAnimationForKey:

Returns the default animation that should be performed for the specified key. (required)

```
+ (id)defaultAnimationForKey:(NSString *)key
```

Parameters

key

The action name or property specified as a string.

Return Value

The animation to perform. A subclass of `CAAnimation`.

Discussion

The `NSAnimatablePropertyContainer` (page 3553) method consults this class method when its search of the receivers `"Getting the Animator Proxy"` (page 3553) dictionary fails to return an animation for *key*.

An animatable property container should implement this method to return a default animation to be performed for each key that it wants to make auto-animatable, where *key* usually references a property of the receiver, but can also specify a special animation trigger (`NSAnimationTriggerOrderIn` (page 3557) or `NSAnimationTriggerOrderOut` (page 3557)).

A developer implementing a custom view subclass, can enable automatic animation for properties by overriding this method, and having it return the desired default `CAAnimation` subclass to use for each of the property keys of interest. The override should defer to super for any keys it doesn't specifically handle, facilitating inheritance of default animation specifications. The following is an example of such an implementation.

```
@implementation MyView
+ (id)defaultAnimationForKey:(NSString *)key {
    if ([key isEqualToString:@"borderColor"]) {
        // By default, animate border color changes with simple linear
interpolation to the new color value.
        return [CABasicAnimation animation];
    } else {
        // Defer to super's implementation for any keys we don't specifically
handle.
```

```
        return [super defaultAnimationForKey:key];
    }
}
@end
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSAnimation.h

Instance Methods

animationForKey:

Returns the animation that should be performed for the specified key. (required)

```
- (id)animationForKey:(NSString *)key
```

Parameters

key
The action name or property specified as a string.

Return Value
The animation to perform. A subclass of `CAAnimation`.

Discussion
When the action specified by *key* is triggered for an object, this method is consulted to find the animation, if any, that should be performed in response.

Like its Core Animation `CALayer` counterpart, `actionForKey:`, this method is a funnel point that defines the order in which the search for an animation proceeds.It first checks the receiver's "Getting the Animator Proxy" (page 3553) dictionary for a value matching *key*, then falls back to `animator` (page 3556) for the receiver's class.

Subclasses should not typically need to override this method.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- animator` (page 3556)
"Managing Animations for Properties" (page 3554)
"Getting the Animator Proxy" (page 3553)

Declared In
NSAnimation.h

animations

Returns the optional dictionary that maps event trigger keys to animation objects. (required)

```
- (NSDictionary *)animations
```

Return Value
The animations as a dictionary.

Discussion
When an action occurs that may trigger an animation the `NSAnimatablePropertyContainer` (page 3553) method first looks in this dictionary for an animation that matches the key. Typically, the key will name a property of the object whose value has just changed, but it may specify a special event trigger (`NSAnimationTriggerOrderIn` (page 3557) or `NSAnimationTriggerOrderOut` (page 3557)).

Availability
Available in Mac OS X v10.5 and later.

See Also
`- animator` (page 3556)
"Managing Animations for Properties" (page 3554)

Declared In
`NSAnimation.h`

animator

Returns a proxy object for the receiver that can be used to initiate implied animation for property changes. (required)

```
- (id)animator
```

Return Value
Returns a proxy object for the receiver that can initiate implied animations in response to property changes.

Discussion
The animator proxy object should be treated as if it was the receiver itself, and may be passed to any code that accepts the receiver as a parameter.

Sending key-value coding compliant "set" messages to the proxy will trigger animation for automatically animated properties of its target object, if the active `NSAnimationContext` in the current thread has a duration value greater than zero, and an animation for the property key is found by the `NSAnimatablePropertyContainer` (page 3553) search mechanism.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
BasicCocoaAnimations
CocoaSlides
LayerBackedOpenGLView
UIElementInspector

Declared In
`NSAnimation.h`

setAnimations:

Sets the option dictionary that maps event trigger keys to animation objects. (required)

```
- (void)setAnimations:(NSDictionary *)dict
```

Parameters

dict

A dictionary containing the event trigger keys and associated animation objects.

Availability

Available in Mac OS X v10.5 and later.

See Also

NSAnimatablePropertyContainer (page 3553)

- animator (page 3556)

"Getting the Animator Proxy" (page 3553)

Declared In

NSAnimation.h

Constants

Transition Animation Keys

The following constants define the keys that reference the transitions used as views are made visible and hidden.

```
NSString *NSAnimationTriggerOrderIn;
NSString *NSAnimationTriggerOrderOut;
```

Constants

NSAnimationTriggerOrderIn

The key that references the transition animation used when a view becomes visible, either as a result of being inserted into the visible view hierarchy or as a result of the view no longer being set as hidden .

Available in Mac OS X v10.5 and later.

Declared in NSAnimation.h.

NSAnimationTriggerOrderOut

The key that references the transition animation used when a view is no longer visible, either as a result of being removed from the visible view hierarchy or as a result of the view set as hidden.

Available in Mac OS X v10.5 and later.

Declared in NSAnimation.h.

NSAnimationDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSAnimation.h
Companion guides	Animation Programming Guide for Cocoa Cocoa Drawing Guide
Related sample code	iSpend

Overview

The `NSAnimationDelegate` protocol defines the optional methods implemented by delegates of `NSAnimation` objects.

Tasks

Controlling and Monitoring an Animation

- `animationDidEnd:` (page 3561)
  Sent to the delegate when the specified animation completes its run.
- `animationDidStop:` (page 3561)
  Sent to the delegate when the specified animation is stopped before it completes its run.
- `animationShouldStart:` (page 3562)
  Sent to the delegate just after an animation is started.
- `animation:valueForProgress:` (page 3560)
  Requests a custom curve value for the current progress value.

Managing Progress Marks

- `animation:didReachProgressMark:` (page 3560)
  Sent to the delegate when an animation reaches a specific progress mark.

Instance Methods

animation:didReachProgressMark:

Sent to the delegate when an animation reaches a specific progress mark.

```
- (void)animation:(NSAnimation *)animation
    didReachProgressMark:(NSAnimationProgress)progress
```

Parameters

animation

A running `NSAnimation` object that has reached a progress mark.

progress

A `float` value (typed as `NSAnimationProgress`) that indicates a progress mark of `animation`.

Discussion

The delegate typically implements this method to perform some animation effect for the time slice indicated by *progress*, such as redrawing objects in a view with new coordinates or changing the frame location or size of a window or view. As an alternative to this delegation message, you may choose to observe the `NSAnimationProgressMarkNotification` (page 116) notification.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSAnimation.h`

animation:valueForProgress:

Requests a custom curve value for the current progress value.

```
- (float)animation:(NSAnimation *)animation
    valueForProgress:(NSAnimationProgress)progress
```

Parameters

animation

An `NSAnimation` object that is running.

progress

A `float` value (typed as `NSAnimationProgress`) that indicates a progress mark of `animation`. This value is always between 0.0 and 1.0.

Return Value

A `float` value representing a custom curve.

Discussion

The delegate can compute and return a custom curve value for the given progress value. If the delegate does not implement this method, `NSAnimation` computes the current curve value.

The `animation:valueForProgress:` message is sent to the delegate when an `NSAnimation` object receives a `currentValue` (page 104) message. The value the delegate returns is used as the value of `currentValue` (page 104); if there is no delegate, or it doesn't implement `animation:valueForProgress:`, `NSAnimation` computes and returns the current value. `NSAnimation` does not invoke `currentValue` (page 104)itself, but subclasses might.

See the description of `currentValue` (page 104) for more information.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `currentValue` (page 104) (NSAnimation)

Declared In
`NSAnimation.h`

animationDidEnd:

Sent to the delegate when the specified animation completes its run.

```
- (void)animationDidEnd:(NSAnimation *)animation
```

Parameters
animation
The `NSAnimation` instance that completed its run.

Discussion
When an `NSAnimation` object reaches the end of its planned duration, it has a progress value of 1.0.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `animationDidStop:` (page 3561)
- `currentProgress` (page 104)

Declared In
`NSAnimation.h`

animationDidStop:

Sent to the delegate when the specified animation is stopped before it completes its run.

```
- (void)animationDidStop:(NSAnimation *)animation
```

Parameters
animation
The `NSAnimation` instance that was stopped.

Discussion
An `NSAnimation` object stops running when it receives a `stopAnimation` (page 112) message.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- animationDidEnd:` (page 3561)

Declared In
`NSAnimation.h`

animationShouldStart:

Sent to the delegate just after an animation is started.

```
- (BOOL)animationShouldStart:(NSAnimation *)animation
```

Parameters

animation
The `NSAnimation` object that was just started.

Return Value
`NO` to cancel the animation, `YES` to have the animation proceed.

Discussion
The delegate is sent this message just after *animation* receives a `startAnimation` (page 111) message. The delegate can use this method to prepare objects and resources for the effect.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- animationDidEnd:` (page 3561)
`- animationDidStop:` (page 3561)

Declared In
`NSAnimation.h`

NSApplicationDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSApplication.h AppKit/NSApplicationScripting.h
Companion guides	Application Architecture Overview Notification Programming Topics Sheet Programming Topics Services Implementation Guide
Related sample code	Cocoa Tips and Tricks Fireworks NineSlice QuickLookDownloader TextSizingExample

Overview

The `NSApplicationDelegate` protocol defines the optional methods implemented by delegates of `NSApplication` objects.

Tasks

Launching Applications

`- applicationWillFinishLaunching:` (page 3578)
Sent by the default notification center immediately before the application object is initialized.

`- applicationDidFinishLaunching:` (page 3571)
Sent by the default notification center after the application has been launched and initialized but before it has received its first event.

Terminating Applications

- `applicationShouldTerminate:` (page 3576)
  Sent to notify the delegate that the application is about to terminate.
- `applicationShouldTerminateAfterLastWindowClosed:` (page 3577)
  Invoked when the user closes the last window the application has open.
- `applicationWillTerminate:` (page 3579)
  Sent by the default notification center immediately before the application terminates.

Managing Active Status

- `applicationWillBecomeActive:` (page 3577)
  Sent by the default notification center immediately before the application becomes active.
- `applicationDidBecomeActive:` (page 3570)
  Sent by the default notification center immediately after the application becomes active.
- `applicationWillResignActive:` (page 3578)
  Sent by the default notification center immediately before the application is deactivated.
- `applicationDidResignActive:` (page 3572)
  Sent by the default notification center immediately after the application is deactivated.

Hiding Applications

- `applicationWillHide:` (page 3578)
  Sent by the default notification center immediately before the application is hidden.
- `applicationDidHide:` (page 3572)
  Sent by the default notification center immediately after the application is hidden.
- `applicationWillUnhide:` (page 3579)
  Sent by the default notification center immediately after the application is unhidden.
- `applicationDidUnhide:` (page 3573)
  Sent by the default notification center immediately after the application is made visible.

Managing Windows

- `applicationWillUpdate:` (page 3580)
  Sent by the default notification center immediately before the application object updates its windows.
- `applicationDidUpdate:` (page 3573)
  Sent by the default notification center immediately after the application object updates its windows.
- `applicationShouldHandleReopen:hasVisibleWindows:` (page 3575)
  Sent by the application to the delegate prior to default behavior to reopen (`rapp`) AppleEvents.

Managing the Dock Menu

- `applicationDockMenu:` (page 3573)
  Allows the delegate to supply a dock menu for the application dynamically.

Displaying Errors

- `application:willPresentError:` (page 3570)
  Sent to the delegate before the specified application presents an error message to the user.

Managing the Screen

- `applicationDidChangeScreenParameters:` (page 3571)
  Sent by the default notification center when the configuration of the displays attached to the computer is changed (either programmatically or when the user changes settings in the Displays control panel).

Opening Files

- `application:openFile:` (page 3566)
  Tells the delegate to open a single file.
- `application:openFileWithoutUI:` (page 3567)
  Tells the delegate to open a file programmatically.
- `application:openTempFile:` (page 3567)
  Tells the delegate to open a temporary file.
- `application:openFiles:` (page 3566)
  Tells the delegate to open multiple files.
- `applicationOpenUntitledFile:` (page 3574)
  Tells the delegate to open an untitled file.
- `applicationShouldOpenUntitledFile:` (page 3575)
  Invoked immediately before opening an untitled file.

Printing

- `application:printFile:` (page 3568)
  Sent when the user starts up the application on the command line with the `-NSPrint` option.
- `application:printFiles:withSettings:showPrintPanels:` (page 3569)
  Prints a group of files.

Instance Methods

application:openFile:

Tells the delegate to open a single file.

```
- (BOOL)application:(NSApplication *)theApplication openFile:(NSString *)filename
```

Parameters

theApplication

The application object associated with the delegate.

filename

The name of the file to open.

Return Value

YES if the file was successfully opened or NO if it was not.

Discussion

Sent directly by *theApplication* to the delegate. The method should open the file *filename*, returning YES if the file is successfully opened, and NO otherwise. If the user started up the application by double-clicking a file, the delegate receives the application:openFile: (page 3566) message before receiving applicationDidFinishLaunching: (page 3571). (applicationWillFinishLaunching: (page 3578) is sent before application:openFile: (page 3566).)

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- application:openFileWithoutUI: (page 3567)
- application:openTempFile: (page 3567)
- applicationOpenUntitledFile: (page 3574)

Related Sample Code

TextSizingExample

Declared In

NSApplication.h

application:openFiles:

Tells the delegate to open multiple files.

```
- (void)application:(NSApplication *)sender openFiles:(NSArray *)filenames
```

Parameters

sender

The application object associated with the delegate.

filenames

An array of NSString objects containing the names of the files to open..

Discussion
Identical to `application:openFile:` (page 3566) except that the receiver opens multiple files corresponding to the file names in the *filenames* array. Delegates should invoke the `replyToOpenOrPrint:` (page 161) method upon success or failure, or when the user cancels the operation.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSApplication.h

application:openFileWithoutUI:

Tells the delegate to open a file programmatically.

```
- (BOOL)application:(id)sender openFileWithoutUI:(NSString *)filename
```

Parameters

sender
The object that sent the command.

filename
The name of the file to open.

Return Value
`YES` if the file was successfully opened or `NO` if it was not.

Discussion
Sent directly by *sender* to the delegate to request that the file *filename* be opened as a linked file. The method should open the file without bringing up its application's user interface—that is, work with the file is under programmatic control of *sender*, rather than under keyboard control of the user.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `application:openFile:` (page 3566)
- `application:openTempFile:` (page 3567)
- `applicationOpenUntitledFile:` (page 3574)
- `application:printFile:` (page 3568)

Declared In
NSApplication.h

application:openTempFile:

Tells the delegate to open a temporary file.

```
- (BOOL)application:(NSApplication *)theApplication openTempFile:(NSString *)filename
```

Parameters

theApplication

The application object associated with the delegate.

filename

The name of the temporary file to open.

Return Value

YES if the file was successfully opened or NO if it was not.

Discussion

Sent directly by *theApplication* to the delegate. The method should attempt to open the file *filename*, returning YES if the file is successfully opened, and NO otherwise.

By design, a file opened through this method is assumed to be temporary—it's the application's responsibility to remove the file at the appropriate time.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- application:openFile: (page 3566)
- application:openFileWithoutUI: (page 3567)
- applicationOpenUntitledFile: (page 3574)

Declared In

NSApplication.h

application:printFile:

Sent when the user starts up the application on the command line with the -NSPrint option.

```
- (BOOL)application:(NSApplication *)theApplication printFile:(NSString *)filename
```

Parameters

theApplication

The application object that is handling the printing.

filename

The name of the file to print.

Return Value

YES if the file was successfully printed or NO if it was not.

Discussion

This message is sent directly by *theApplication* to the delegate. The application terminates (using the terminate: (page 176) method) after this method returns.

If at all possible, this method should print the file without displaying the user interface. For example, if you pass the -NSPrint option to the TextEdit application, TextEdit assumes you want to print the entire contents of the specified file. However, if the application opens more complex documents, you may want to display a panel that lets the user choose exactly what they want to print.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- application:openFileWithoutUI:` (page 3567)

Declared In
`NSApplication.h`

application:printFiles:withSettings:showPrintPanels:

Prints a group of files.

```
- (NSApplicationPrintReply)application:(NSApplication *)application
    printFiles:(NSArray *)fileNames withSettings:(NSDictionary *)printSettings
    showPrintPanels:(BOOL)showPrintPanels
```

Parameters

application
The application object that is handling the printing.

fileNames
An array of `NSString` objects, each of which contains the name of a file to print.

printSettings
A dictionary containing `NSPrintInfo`-compatible print job attributes.

showPrintPanels
A Boolean that specifies whether the print panel should be displayed for each file printed. Print progress indicators will be presented even if this value is `NO`.

Return Value
A constant indicating whether printing was successful. For a list of possible values, see "NSApplicationPrintReply" (page 189).

Discussion
Return `NSPrintingReplyLater` if the result of printing cannot be returned immediately, for example, if printing will cause the presentation of a sheet. If your method returns `NSPrintingReplyLater` it must always invoke the `NSApplication` method `replyToOpenOrPrint:` (page 161)] when the entire print operation has been completed, successfully or not.

This delegate method replaces `application:printFiles:`, which is now deprecated. If your application delegate only implements the deprecated method, it is still invoked, and `NSApplication` uses private functionality to arrange for the print settings to take effect.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSApplication.h`

application:willPresentError:

Sent to the delegate before the specified application presents an error message to the user.

```
- (NSError *)application:(NSApplication *)application willPresentError:(NSError
    *)error
```

Parameters

application

The application object associated with the delegate.

error

The error object that is used to construct the error message. Your implementation of this method can return a new `NSError` object or the same one in this parameter.

Return Value
The error object to display.

Discussion
You can implement this delegate method to customize the presentation of any error presented by your application, as long as no code in your application overrides either of the `NSResponder` methods `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` or `presentError:` in a way that prevents errors from being passed down the responder chain to the application object.

Your implementation of this delegate method can examine *error* and, if its localized description or recovery information is unhelpfully generic, return an error object with specific localized text that is more suitable for presentation in alert sheets and dialogs. If you do this, always use the domain and error code of the `NSError` object to distinguish between errors whose presentation you want to customize and those you do not. Don't make decisions based on the localized description, recovery suggestion, or recovery options because parsing localized text is problematic. If you decide not to customize the error presentation, just return the passed-in error object.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSApplication.h`

applicationDidBecomeActive:

Sent by the default notification center immediately after the application becomes active.

```
- (void)applicationDidBecomeActive:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationDidBecomeActiveNotification` (page 193). Calling the `object` method of this notification returns the `NSApplication` object itself.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `applicationDidFinishLaunching:` (page 3571)
- `applicationDidResignActive:` (page 3572)
- `applicationWillBecomeActive:` (page 3577)

Declared In
`NSApplication.h`

applicationDidChangeScreenParameters:

Sent by the default notification center when the configuration of the displays attached to the computer is changed (either programmatically or when the user changes settings in the Displays control panel).

```
- (void)applicationDidChangeScreenParameters:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationDidChangeScreenParametersNotification` (page 194). Calling the `object` method of this notification returns the `NSApplication` object itself.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSApplication.h`

applicationDidFinishLaunching:

Sent by the default notification center after the application has been launched and initialized but before it has received its first event.

```
- (void)applicationDidFinishLaunching:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationDidFinishLaunchingNotification` (page 194). Calling the `object` method of this notification returns the `NSApplication` object itself.

Discussion
Delegates can implement this method to perform further initialization. This method is called after the application's main run loop has been started but before it has processed any events. If the application was launched by the user opening a file, the delegate's `application:openFile:` method is called before this method. If you want to perform initialization before any files are opened, implement the `applicationWillFinishLaunching:` (page 3578) method in your delegate, which is called before `application:openFile:`.)

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `finishLaunching` (page 147) (NSApplication)
- `applicationWillFinishLaunching:` (page 3578)
- `applicationDidBecomeActive:` (page 3570)
- `application:openFile:` (page 3566)

Declared In
`NSApplication.h`

applicationDidHide:

Sent by the default notification center immediately after the application is hidden.

```
- (void)applicationDidHide:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationDidHideNotification` (page 194). Calling the `object` method of this notification returns the `NSApplication` object itself.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `applicationWillHide:` (page 3578)
- `applicationDidHide:` (page 3572)
- `unhide:` (page 177) (NSApplication)

Declared In
`NSApplication.h`

applicationDidResignActive:

Sent by the default notification center immediately after the application is deactivated.

```
- (void)applicationDidResignActive:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationDidResignActiveNotification` (page 194). Calling the `object` method of this notification returns the `NSApplication` object itself.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `applicationDidBecomeActive:` (page 3570)
- `applicationWillResignActive:` (page 3578)

Declared In
NSApplication.h

applicationDidUnhide:

Sent by the default notification center immediately after the application is made visible.

```
- (void)applicationDidUnhide:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationDidUnhideNotification` (page 195). Calling the `object` method of this notification returns the `NSApplication` object itself.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `applicationDidHide:` (page 3572)
- `applicationWillUnhide:` (page 3579)
- `unhide:` (page 177) (NSApplication)

Declared In
NSApplication.h

applicationDidUpdate:

Sent by the default notification center immediately after the application object updates its windows.

```
- (void)applicationDidUpdate:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationDidUpdateNotification` (page 195). Calling the `object` method of this notification returns the `NSApplication` object itself.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `applicationWillUpdate:` (page 3580)
- `updateWindows` (page 179) (NSApplication)

Declared In
NSApplication.h

applicationDockMenu:

Allows the delegate to supply a dock menu for the application dynamically.

```
- (NSMenu *)applicationDockMenu:(NSApplication *)sender
```

Parameters

sender

The application object associated with the delegate.

Return Value

The menu to display in the dock.

Discussion

You can also connect a menu in Interface Builder to the `dockMenu` outlet. A third way for your application to specify a dock menu is to provide an `NSMenu` in a nib.

If this method returns a menu, this menu takes precedence over the `dockMenu` in the nib.

The target and action for each menu item are passed to the dock. On selection of the menu item the dock messages your application, which should invoke `[NSApp sendAction:selector to:target from:nil]`.

To specify an `NSMenu` in a nib, you add the nib name to the `info.plist`, using the key `AppleDockMenu`. The nib name is specified without an extension. You then create a connection from the file's owner object (which by default is `NSApplication`) to the menu. Connect the menu to the `dockMenu` outlet of `NSApplication`. The menu is in its own nib file so it can be loaded lazily when the `dockMenu` is requested, rather than at launch time.

Availability

Available in Mac OS X v10.1 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSApplication.h`

applicationOpenUntitledFile:

Tells the delegate to open an untitled file.

```
- (BOOL)applicationOpenUntitledFile:(NSApplication *)theApplication
```

Parameters

theApplication

The application object associated with the delegate.

Return Value

`YES` if the file was successfully opened or `NO` if it was not.

Discussion

Sent directly by *theApplication* to the delegate to request that a new, untitled file be opened.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- `application:openFile:` (page 3566)
- `application:openFileWithoutUI:` (page 3567)
- `application:openTempFile:` (page 3567)

Declared In
NSApplication.h

applicationShouldHandleReopen:hasVisibleWindows:

Sent by the application to the delegate prior to default behavior to reopen (rapp) AppleEvents.

```
- (BOOL)applicationShouldHandleReopen:(NSApplication *)theApplication
    hasVisibleWindows:(BOOL)flag
```

Parameters

theApplication
The application object.

flag
Indicates whether the NSApplication object found any visible windows in your application. You can use this value as an indication of whether the application would do anything if you return YES.

Return Value
YES if you want the application to perform its normal tasks or NO if you want the application to do nothing.

Discussion
These events are sent whenever the Finder reactivates an already running application because someone double-clicked it again or used the dock to activate it.

By default the Application Kit will handle this event by checking whether there are any visible NSWindow (not NSPanel) objects, and, if there are none, it goes through the standard untitled document creation (the same as it does if *theApplication* is launched without any document to open). For most document-based applications, an untitled document will be created.

The application delegate will also get a chance to respond to the normal untitled document delegate methods. If you implement this method in your application delegate, it will be called before any of the default behavior happens. If you return YES, then NSApplication will proceed as normal. If you return NO, then NSApplication will do nothing. So, you can either implement this method with a version that does nothing, and return NO if you do not want anything to happen at all (not recommended), or you can implement this method, handle the event yourself in some custom way, and return NO.

Miniaturized windows, windows in the Dock, are considered visible by this method, and cause *flag* to return YES, despite the fact that miniaturized windows return NO when sent an isVisible (page 3342) message.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSApplication.h

applicationShouldOpenUntitledFile:

Invoked immediately before opening an untitled file.

```
- (BOOL)applicationShouldOpenUntitledFile:(NSApplication *)sender
```

Parameters

sender

The application object associated with the delegate.

Return Value

YES if the application should open a new untitled file or NO if it should not.

Discussion

Use this method to decide whether the application should open a new, untitled file. Note that applicationOpenUntitledFile: (page 3574) is invoked if this method returns YES.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSApplication.h

applicationShouldTerminate:

Sent to notify the delegate that the application is about to terminate.

```
- (NSApplicationTerminateReply)applicationShouldTerminate:(NSApplication *)sender
```

Parameters

sender

The application object that is about to be terminated.

Return Value

One of the values defined in NSApplicationTerminateReply (page 189) constants indicating whether the application should terminate. For compatibility reasons, a return value of NO is equivalent to NSTerminateCancel (page 189), and a return value of YES is equivalent to NSTerminateNow (page 189).

Discussion

This method is called after the application's Quit menu item has been selected, or after the terminate: (page 176) method has been called. Generally, you should return NSTerminateNow (page 189) to allow the termination to complete, but you can cancel the termination process or delay it somewhat as needed. For example, you might delay termination to finish processing some critical data but then terminate the application as soon as you are done by calling the replyToApplicationShouldTerminate: (page 160) method.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- terminate: (page 176) (NSApplication)
- applicationShouldTerminateAfterLastWindowClosed: (page 3577)
- applicationWillTerminate: (page 3579)

Declared In

NSApplication.h

applicationShouldTerminateAfterLastWindowClosed:

Invoked when the user closes the last window the application has open.

```
- (BOOL)applicationShouldTerminateAfterLastWindowClosed:(NSApplication
    *)theApplication
```

Parameters

theApplication

The application object whose last window was closed.

Return Value

`NO` if the application should not be terminated when its last window is closed; otherwise, `YES` to terminate the application.

Discussion

The application sends this message to your delegate when the application's last window is closed. It sends this message regardless of whether there are still panels open. (A panel in this case is defined as being an instance of `NSPanel` or one of its subclasses.)

If your implementation returns `NO`, control returns to the main event loop and the application is not terminated. If you return `YES`, your delegate's `applicationShouldTerminate:` method is subsequently invoked to confirm that the application should be terminated.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- `terminate:` (page 176)
- `applicationShouldTerminate:` (page 3576)

Declared In

`NSApplication.h`

applicationWillBecomeActive:

Sent by the default notification center immediately before the application becomes active.

```
- (void)applicationWillBecomeActive:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationWillBecomeActiveNotification` (page 195). Calling the `object` method of this notification returns the `NSApplication` object itself.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- `applicationDidBecomeActive:` (page 3570)
- `applicationWillFinishLaunching:` (page 3578)
- `applicationWillResignActive:` (page 3578)

Declared In
NSApplication.h

applicationWillFinishLaunching:

Sent by the default notification center immediately before the application object is initialized.

```
- (void)applicationWillFinishLaunching:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named NSApplicationWillFinishLaunchingNotification (page 195). Calling the object method of this notification returns the NSApplication object itself.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- applicationDidFinishLaunching: (page 3571)
- applicationWillBecomeActive: (page 3577)
- finishLaunching (page 147) (NSApplication)

Declared In
NSApplication.h

applicationWillHide:

Sent by the default notification center immediately before the application is hidden.

```
- (void)applicationWillHide:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named NSApplicationWillHideNotification (page 195). Calling the object method of this notification returns the NSApplication object itself.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- applicationDidHide: (page 3572)
- hide: (page 148) (NSApplication)

Declared In
NSApplication.h

applicationWillResignActive:

Sent by the default notification center immediately before the application is deactivated.

```
- (void)applicationWillResignActive:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationWillResignActiveNotification` (page 196). Calling the `object` method of this notification returns the `NSApplication` object itself.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- `applicationWillBecomeActive:` (page 3577)
- `applicationDidResignActive:` (page 3572)

Declared In

`NSApplication.h`

applicationWillTerminate:

Sent by the default notification center immediately before the application terminates.

```
- (void)applicationWillTerminate:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationWillTerminateNotification` (page 196). Calling the `object` method of this notification returns the `NSApplication` object itself.

Discussion

Your delegate can use this method to perform any final cleanup before the application terminates.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- `applicationShouldTerminate:` (page 3576)
- `terminate:` (page 176) (NSApplication)

Declared In

`NSApplication.h`

applicationWillUnhide:

Sent by the default notification center immediately after the application is unhidden.

```
- (void)applicationWillUnhide:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationWillUnhideNotification` (page 196). Calling the `object` method of this notification returns the `NSApplication` object itself.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- `unhide:` (page 177) (NSApplication)
- `applicationDidUnhide:` (page 3573)
- `applicationWillHide:` (page 3578)

Declared In

`NSApplication.h`

applicationWillUpdate:

Sent by the default notification center immediately before the application object updates its windows.

```
- (void)applicationWillUpdate:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSApplicationWillUpdateNotification` (page 196). Calling the `object` method of this notification returns the `NSApplication` object itself.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- `applicationDidUpdate:` (page 3573)
- `updateWindows` (page 179) (NSApplication)

Declared In

`NSApplication.h`

NSBrowserDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSBrowser.h
Companion guide	Browsers
Related sample code	AnimatedTableView

Overview

The `NSBrowserDelegate` protocol defines the method that a delegate of `NSBrowser` should implement.

Tasks

Getting Browser Information

- `browser:isColumnValid:` (page 3589)
  Returns whether the contents of the specified column are valid.
- `browser:numberOfRowsInColumn:` (page 3591)
  Returns the number of rows of data in the specified column.
- `browser:numberOfChildrenOfItem:` (page 3591)
  Asks the delegate for the number of children the given item has.
- `browser:titleOfColumn:` (page 3598)
  Asks the delegate for the title to display above the specified column.

Managing Selection Behavior

- `browser:shouldTypeSelectForEvent:withCurrentSearchString:` (page 3597)
  Sent to the delegate to determine whether keyboard-based selection (type select) for a given event and search string should proceed.

- `browser:typeSelectStringForRow:inColumn:` (page 3599)
  Sent to the delegate to get the keyboard-based selection (type select) string for the specified row and column.
- `browser:nextTypeSelectMatchFromRow:toRow:inColumn:forString:` (page 3590)
  Sent to the delegate to customize a browser's keyboard-based selection (type select) behavior.

Managing Selection

- `browser:selectCellWithString:inColumn:` (page 3593)
  Asks the delegate to select the cell with the given title in the specified column.
- `browser:selectRow:inColumn:` (page 3594)
  Asks the delegate to select the cell at the specified row and column location.
- `browser:selectionIndexesForProposedSelection:inColumn:` (page 3594)
  Asks the delegate for a set of indexes to select when the user changes the selection in the browser with the keyboard or mouse.

Accessing Components

- `browser:child:ofItem:` (page 3585)
  Asks the delegate to return the child of the specified item at the specified index.
- `browser:isLeafItem:` (page 3589)
  Asks the delegate whether the specified item is a leaf item (an item that cannot be expanded).
- `browser:shouldEditItem:` (page 3595)
  Asks the delegate whether the browser may start an editing session for the specified item.
- `browser:objectValueForItem:` (page 3592)
  Returns the object that the specified item uses to draw its contents.
- `browser:setObjectValue:forItem:` (page 3595)
  Sets the object that the specified item uses to draw its contents to the specified object.
- `rootItemForBrowser:` (page 3603)
  Asks the delegate to return the root item of the browser.
- `browser:previewViewControllerForLeafItem:` (page 3592)
  Asks the delegate for a controller that provides a preview column for the specified leaf item.
- `browser:headerViewControllerForItem:` (page 3587)
  Asks the delegate for a controller that provides a header view for the specified column item.

Managing Columns

- `browser:createRowsForColumn:inMatrix:` (page 3586)
  Creates a row in the given matrix for each row of data in the specified column of the browser.
- `browser:willDisplayCell:atRow:column:` (page 3601)
  Gives the delegate the opportunity to modify the specified cell at the given row and column location before the browser displays it.

- `browser:didChangeLastColumn:toColumn:` (page 3586)
  Tells the delegate that the browser's last column changed.

Scrolling

- `browserWillScroll:` (page 3603)
  Notifies the delegate when the browser will scroll.
- `browserDidScroll:` (page 3603)
  Notifies the delegate when the browser has scrolled.

Dragging

- `browser:canDragRowsWithIndexes:inColumn:withEvent:` (page 3585)
  Sent to the delegate to determine whether the browser can attempt to initiate a drag of the specified rows for the specified event.
- `browser:draggingImageForRowsWithIndexes:inColumn:withEvent:offset:` (page 3587)
  Sent to the delegate to obtain an image to represent dragged rows during a drag operation on a browser.
- `browser:validateDrop:proposedRow:column:dropOperation:` (page 3600)
  Sent to the delegate during a dragging session to determine whether a drop should be accepted and to obtain the drop location. This method is required for a browser to be a drag destination.
- `browser:acceptDrop:atRow:column:dropOperation:` (page 3584)
  Sent to the delegate during a dragging session to determine whether to accept the drop.
- `browser:writeRowsWithIndexes:inColumn:toPasteboard:` (page 3602)
  Determines whether a drag operation can proceed. This method is required for a browser to be a drag source.
- `browser:namesOfPromisedFilesDroppedAtDestination:forDraggedRowsWithIndexes:inColumn:` (page 3589)
  Implements file promise drag operations.

Sizing

- `browser:shouldSizeColumn:forUserResize:toWidth:` (page 3596)
  Used to determine a column's initial size.
- `browser:sizeToFitWidthOfColumn:` (page 3598)
  Returns the ideal width for a column.
- `browserColumnConfigurationDidChange:` (page 3602)
  Used by clients to implement their own column width persistence.
- `browser:heightOfRow:inColumn:` (page 3588)
  Specifies the height of the specified row in the specified column.

Displaying Cell Content

`- browser:shouldShowCellExpansionForRow:column:` (page 3596)
Invoked to allow the delegate to control cell expansion for a specific row and column.

Instance Methods

browser:acceptDrop:atRow:column:dropOperation:

Sent to the delegate during a dragging session to determine whether to accept the drop.

```
- (BOOL)browser:(NSBrowser *)browser
    acceptDrop:(id <NSDraggingInfo>)info
    atRow:(NSInteger)row
    column:(NSInteger)column
    dropOperation:(NSBrowserDropOperation)dropOperation
```

Parameters

browser
The browser.

info
The drag session information.

row
The drop row.

column
The drop column.

dropOperation
The drop location relative to *row*.

Return Value
`YES` to accept the drop; `NO` to decline it.

Discussion
This method is required for a browser to be a drag destination. It is invoked after the `browser:validateDrop:proposedRow:column:dropOperation:` (page 3600) method allows the drop.

The delegate should incorporate the pasteboard data from the dragging session (*info*`.draggingPasteboard`).

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSBrowser.h`

browser:canDragRowsWithIndexes:inColumn:withEvent:

Sent to the delegate to determine whether the browser can attempt to initiate a drag of the specified rows for the specified event.

```
- (BOOL)browser:(NSBrowser *)browser
    canDragRowsWithIndexes:(NSIndexSet *)rowIndexes
    inColumn:(NSInteger)column
    withEvent:(NSEvent *)event
```

Parameters

browser
The browser.

rowIndexes
The rows the user is dragging.

column
The column containing the rows the user is dragging.

event
The drag event.

Return Value
`YES` to allow the drag operation; `NO` to disallow it.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- canDragRowsWithIndexes:inColumn:withEvent:` (page 407) (`NSBrowser`)

Declared In
`NSBrowser.h`

browser:child:ofItem:

Asks the delegate to return the child of the specified item at the specified index.

```
- (id)browser:(NSBrowser *)browser
    child:(NSInteger)index
    ofItem:(id)item
```

Parameters

browser
The browser.

index
The child's index.

item
The item containing the child.

Return Value
The child at the specified index.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSBrowser.h

browser:createRowsForColumn:inMatrix:

Creates a row in the given matrix for each row of data in the specified column of the browser.

```
- (void)browser:(NSBrowser *)sender
    createRowsForColumn:(NSInteger)column
    inMatrix:(NSMatrix *)matrix
```

Parameters

sender
The browser.

column
The index of the column in which the rows are located.

matrix
The matrix in which the rows are created.

Discussion
Either this method or `browser:numberOfRowsInColumn:` (page 3591) must be implemented, but not both, or an `NSBrowserIllegalDelegateException` will be raised.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- browser:willDisplayCell:atRow:column:` (page 3601)

Declared In
NSBrowser.h

browser:didChangeLastColumn:toColumn:

Tells the delegate that the browser's last column changed.

```
- (void)browser:(NSBrowser *)browser
    didChangeLastColumn:(NSInteger)oldLastColumn
    toColumn:(NSInteger)column
```

Parameters

browser
The browser.

oldLastColumn
The index of the old last column.

column
The index of the new last column.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSBrowser.h

browser:draggingImageForRowsWithIndexes:inColumn:withEvent:offset:

Sent to the delegate to obtain an image to represent dragged rows during a drag operation on a browser.

```
- (NSImage *)browser:(NSBrowser *)browser
    draggingImageForRowsWithIndexes:(NSIndexSet *)rowIndexes
    inColumn:(NSInteger)column
    withEvent:(NSEvent *)event
    offset:(NSPointPointer)dragImageOffset
```

Parameters

browser
The browser.

rowIndexes
The indexes of the rows the user is dragging.

column
The column containing the rows the user is dragging.

event
The drag event.

dragImageOffset
The offset for the returned image:

- NSZeroPoint: Centers the image under the pointer.

Return Value
An image representing the visible rows identified by *rowIndexes*.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- draggingImageForRowsWithIndexes:inColumn:withEvent:offset: (page 413) (NSBrowser)

Declared In
NSBrowser.h

browser:headerViewControllerForItem:

Asks the delegate for a controller that provides a header view for the specified column item.

```
- (NSViewController *)browser:(NSBrowser *)browser
    headerViewControllerForItem:(id)item
```

Parameters

browser

The browser.

item

The column item.

Return Value

A view controller that provides a header view, or `nil` to omit the header view.

Discussion

The returned controller's represented object will be set to the column item. This method is called only if the delegate implements the item data source methods.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- browser:previewViewControllerForLeafItem:` (page 3592)

Declared In

`NSBrowser.h`

browser:heightOfRow:inColumn:

Specifies the height of the specified row in the specified column.

```
- (CGFloat)browser:(NSBrowser *)browser
    heightOfRow:(NSInteger)row
    inColumn:(NSInteger)columnIndex
```

Parameters

browser

The browser.

row

The index of the row.

columnIndex

The index of the column.

Return Value

The height to set for the specified row, which must be greater than 0.

Discussion

The values returned for this method may be cached. Therefore, you should call `noteHeightOfRowsWithIndexesChanged:inColumn:` (page 423) to invalidate a row's height before changing it.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSBrowser.h`

browser:isColumnValid:

Returns whether the contents of the specified column are valid.

```
- (BOOL)browser:(NSBrowser *)sender
    isColumnValid:(NSInteger)column
```

Parameters

sender

The browser containing the column to validate.

column

The index of the column to validate.

Return Value

`YES` if the column's contents are valid; otherwise, `NO`. If `NO` is returned, *sender* reloads the column.

Discussion

This method is invoked in response to the `validateVisibleColumns` (page 451)method of `NSBrowser` being sent to *sender*.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSBrowser.h`

browser:isLeafItem:

Asks the delegate whether the specified item is a leaf item (an item that cannot be expanded).

```
- (BOOL)browser:(NSBrowser *)browser
    isLeafItem:(id)item
```

Parameters

browser

The browser.

item

The item to be checked.

Return Value

`YES` if the specified item is a leaf item; otherwise, `NO`.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSBrowser.h`

browser:namesOfPromisedFilesDroppedAtDestination:forDraggedRowsWithIndexes:inColumn:

Implements file promise drag operations.

```
- (NSArray *)browser:(NSBrowser *)browser
    namesOfPromisedFilesDroppedAtDestination:(NSURL *)dropDestination
    forDraggedRowsWithIndexes:(NSIndexSet *)rowIndexes
    inColumn:(NSInteger)column
```

Parameters

browser

The browser.

dropDestination

The drop filesystem location.

rowIndexes

The indexes of the rows the user is dropping.

column

The index of the column containing the rows the user is dropping.

Return Value

Filenames (not pathnames) for the actual files represented by the rows the user is dropping.

Discussion

Note that file promise drag operation support requires adding the data type `NSFilesPromisePboardType` to the pasteboard in the `browser:writeRowsWithIndexes:inColumn:toPasteboard:` (page 3602) method.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

`- namesOfPromisedFilesDroppedAtDestination:` (page 423) (`NSBrowser`)

Related Sample Code

ZipBrowser

Declared In

`NSBrowser.h`

browser:nextTypeSelectMatchFromRow:toRow:inColumn:forString:

Sent to the delegate to customize a browser's keyboard-based selection (type select) behavior.

```
- (NSInteger)browser:(NSBrowser *)browser
    nextTypeSelectMatchFromRow:(NSInteger)startRow
    toRow:(NSInteger)endRow
    inColumn:(NSInteger)column
    forString:(NSString *)searchString
```

Parameters

browser

The browser.

startRow

The beginning of the row set to search.

endRow
The end of the row set to search. This value can be less than *startRowIndex* when the search wraps around to the beginning.

column
The column containing the rows being searched.

searchString
The keyboard-based selection string. It is `nil` when no keyboard-based selection has begun.

Return Value
The index of the first row that matches *searchString* between *startRowIndex* and *endRowIndex* - 1, or `-1` if there is no match.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `browser:shouldTypeSelectForEvent:withCurrentSearchString:` (page 3597)
- `browser:typeSelectStringForRow:inColumn:` (page 3599)

Declared In
`NSBrowser.h`

browser:numberOfChildrenOfItem:

Asks the delegate for the number of children the given item has.

```
- (NSInteger)browser:(NSBrowser *)browser
    numberOfChildrenOfItem:(id)item
```

Parameters

browser
The browser.

item
The item that has some number of children.

Return Value
The number of children.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSBrowser.h`

browser:numberOfRowsInColumn:

Returns the number of rows of data in the specified column.

```
- (NSInteger)browser:(NSBrowser *)sender
    numberOfRowsInColumn:(NSInteger)column
```

Parameters

sender

The browser.

column

The index of the column.

Return Value

The number of rows of data.

Discussion

Either this method or `browser:createRowsForColumn:inMatrix:` (page 3586) must be implemented, but not both.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

`- browser:willDisplayCell:atRow:column:` (page 3601)

Declared In

`NSBrowser.h`

browser:objectValueForItem:

Returns the object that the specified item uses to draw its contents.

```
- (id)browser:(NSBrowser *)browser
    objectValueForItem:(id)item
```

Parameters

browser

The browser.

item

The item in question.

Return Value

The item's object.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- browser:setObjectValue:forItem:` (page 3595)

Declared In

`NSBrowser.h`

browser:previewViewControllerForLeafItem:

Asks the delegate for a controller that provides a preview column for the specified leaf item.

```
- (NSViewController *)browser:(NSBrowser *)browser
    previewViewControllerForLeafItem:(id)item
```

Parameters

browser

The browser.

item

The leaf item.

Return Value

A view controller that provides a preview column, or `nil` to suppress the preview column.

Discussion

The returned controller's represented object is set to the specified leaf item. This method is called only if the delegate implements the item data source methods.

Availability

Available in Mac OS X v10.6 and later.

See Also

`- browser:headerViewControllerForItem:` (page 3587)

Declared In

`NSBrowser.h`

browser:selectCellWithString:inColumn:

Asks the delegate to select the cell with the given title in the specified column.

```
- (BOOL)browser:(NSBrowser *)sender
    selectCellWithString:(NSString *)title
    inColumn:(NSInteger)column
```

Parameters

sender

The browser.

title

The title of the cell to select.

column

The index of the column containing the cell to select.

Return Value

`YES` if the cell was successfully selected; otherwise, `NO`.

Discussion

Invoked in response to the `setPath:` (page 443) method of `NSBrowser` being received by *sender*.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

`- selectedCellInColumn:` (page 430) (`NSBrowser`)

Declared In
NSBrowser.h

browser:selectionIndexesForProposedSelection:inColumn:

Asks the delegate for a set of indexes to select when the user changes the selection in the browser with the keyboard or mouse.

```
- (NSIndexSet *)browser:(NSBrowser *)browser
    selectionIndexesForProposedSelection:(NSIndexSet *)proposedSelectionIndexes
    inColumn:(NSInteger)column
```

Parameters

browser
The browser.

proposedSelectionIndexes
The set of indexes of the items in the proposed selection.

column
The column index of the column containing the selection.

Return Value
The set of indexes of the items that should be selected.

Discussion
This method may be called multiple times, with one new index added to the previous selection, to see whether a particular index can be selected when the user is extending the selection with the keyboard or mouse. The *proposedSelectionIndexes* parameter contains the entire selection, and you can return the existing selection if you do not want to change it. This method works only for item-based browsers.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSBrowser.h

browser:selectRow:inColumn:

Asks the delegate to select the cell at the specified row and column location.

```
- (BOOL)browser:(NSBrowser *)sender
    selectRow:(NSInteger)row
    inColumn:(NSInteger)column
```

Parameters

sender
The browser.

row
The index of the row containing the cell to select.

column
The index of the column containing the cell to select.

Return Value
`YES` if the cell was selected; otherwise, `NO`.

Discussion
Invoked in response to `selectRow:inColumn:` (page 433) of `NSBrowser` being received by *sender*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `selectedRowInColumn:` (page 432) (`NSBrowser`)
- `selectRow:inColumn:` (page 433) (`NSBrowser`)

Declared In
`NSBrowser.h`

browser:setObjectValue:forItem:

Sets the object that the specified item uses to draw its contents to the specified object.

```
- (void)browser:(NSBrowser *)browser
    setObjectValue:(id)object
    forItem:(id)item
```

Parameters

browser
The browser.

object
The object to set.

item
The item whose object is set.

Availability
Available in Mac OS X v10.6 and later.

See Also
- `browser:objectValueForItem:` (page 3592)

Declared In
`NSBrowser.h`

browser:shouldEditItem:

Asks the delegate whether the browser may start an editing session for the specified item.

```
- (BOOL)browser:(NSBrowser *)browser
    shouldEditItem:(id)item
```

Parameters

browser
The browser.

item
The item to edit.

Return Value
YES to allow the editing session to begin; NO to disallow it.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSBrowser.h

browser:shouldShowCellExpansionForRow:column:

Invoked to allow the delegate to control cell expansion for a specific row and column.

```
- (BOOL)browser:(NSBrowser *)browser
    shouldShowCellExpansionForRow:(NSInteger)row
    column:(NSInteger)column
```

Parameters
browser
The browser.

row
The index of the row requesting an expansion tooltip.

column
The index of the column containing the requesting row.

Return Value
YES to allow the cell expansion tooltip; NO to disallow it.

Discussion
Cell expansion can occur when the mouse hovers over the specified cell and the cell contents are unable to be fully displayed within the cell. If this method returns YES, the full cell contents will be shown in a special floating tool tip view, otherwise the content is truncated.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSBrowser.h

browser:shouldSizeColumn:forUserResize:toWidth:

Used to determine a column's initial size.

```
- (CGFloat)browser:(NSBrowser *)browser
    shouldSizeColumn:(NSInteger)columnIndex
    forUserResize:(BOOL)forUserResize
    toWidth:(CGFloat)suggestedWidth
```

Parameters

browser
The browser.

columnIndex
The index of the column to size.

forUserResize
Currently, this is always set to NO.

suggestedWidth
The suggested width for the column.

Return Value
The delegate's desired initial width for a newly added column. If you want to accept the suggested width, return *suggestedWidth*. If you return 0 or a size too small to display the resize handle and a portion of the column, the actual size used will be larger than the size you requested.

Discussion
This method applies only to browsers with resize type NSBrowserNoColumnResizing (page 452) or NSBrowserUserColumnResizing (page 452) (see NSBrowserColumnResizingType (page 451)).

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- setWidth:ofColumn: (page 448) (NSBrowser)

Declared In
NSBrowser.h

browser:shouldTypeSelectForEvent:withCurrentSearchString:

Sent to the delegate to determine whether keyboard-based selection (type select) for a given event and search string should proceed.

```
- (BOOL)browser:(NSBrowser *)browser
    shouldTypeSelectForEvent:(NSEvent *)event
    withCurrentSearchString:(NSString *)searchString
```

Parameters

browser
The browser.

event
The keyboard event being processed.

searchString
The keyboard-based selection string. It is nil when no keyboard-based selection has begun.

Return Value
YES to allow the selection; NO to disallow it.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `allowsTypeSelect` (page 405) (`NSBrowser`)
- `browser:nextTypeSelectMatchFromRow:toRow:inColumn:forString:` (page 3590)
- `browser:typeSelectStringForRow:inColumn:` (page 3599)

Declared In
`NSBrowser.h`

browser:sizeToFitWidthOfColumn:

Returns the ideal width for a column.

```
- (CGFloat)browser:(NSBrowser *)browser
    sizeToFitWidthOfColumn:(NSInteger)columnIndex
```

Parameters

browser
The browser.

columnIndex
The index of the column to size. If `-1`, the result is used to resize all columns.

Return Value
The ideal width of the column. This method is used when performing a "right-size" operation, that is, when sizing a column to the smallest width that contains all the content without clipping or truncating.

If *columnIndex* is `-1`, you should return a size that can be uniformly applied to all columns (that is, every column will be set to this size).

Returning a value of `-1` allows you to opt-out of providing a width for the requested column.

Discussion
This method applies only to browsers with resize type `NSBrowserUserColumnResizing`.

It is assumed that the implementation may be expensive, so it will be called only when necessary.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSBrowser.h`

browser:titleOfColumn:

Asks the delegate for the title to display above the specified column.

```
- (NSString *)browser:(NSBrowser *)sender
    titleOfColumn:(NSInteger)column
```

Parameters

sender
The browser.

column
The index the column to be titled.

Return Value
The title of the specified column.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `setTitle:ofColumn:` (page 447)
- `titleOfColumn:` (page 450)

Declared In
`NSBrowser.h`

browser:typeSelectStringForRow:inColumn:

Sent to the delegate to get the keyboard-based selection (type select) string for the specified row and column.

```
- (NSString *)browser:(NSBrowser *)browser
    typeSelectStringForRow:(NSInteger)row
    inColumn:(NSInteger)column
```

Parameters
browser
The browser.

row
The row index.

column
The column index.

Return Value
The keyboard-based selection string.

Discussion
Returning the empty string or `nil` (for example, when the cell does not contain text) specifies that the [*column*, *row*] cell has no text to search.

If the delegate does not implement this method, all cells with text are searched, and the browser determines the keyboard-based selection text by sending `stringValue` (page 605) to the cell specified by *column* and *row*.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `browser:shouldTypeSelectForEvent:withCurrentSearchString:` (page 3597)
- `browser:nextTypeSelectMatchFromRow:toRow:inColumn:forString:` (page 3590)

Declared In
NSBrowser.h

browser:validateDrop:proposedRow:column:dropOperation:

Sent to the delegate during a dragging session to determine whether a drop should be accepted and to obtain the drop location. This method is required for a browser to be a drag destination.

```
- (NSDragOperation)browser:(NSBrowser *)browser
    validateDrop:(id <NSDraggingInfo>)info
    proposedRow:(NSInteger *)row
    column:(NSInteger *)column
    dropOperation:(NSBrowserDropOperation *)dropOperation
```

Parameters

browser
The browser.

info
The drag session information.

row
On input, the proposed drop row. On output, the drop row.

column
On input, the proposed drop column. On output, the drop column.

dropOperation
On input, the proposed drop location. On output, the drop location.

Return Value
The drag operation that the data source is to perform. For the browser to accept the drop, it must not be NSDragOperationNone (page 3666).

Discussion
The browser proposes a drop column, row, and row-relative location for the drop based on the pointer position, as shown in this table:

Drop relative location	Description
NSBrowserDropOn (page 452)	Dragging location (*dragInfo*.draggingLocation) is closer to the middle of *row* than to either of its vertical sides.
NSBrowserDropAbove (page 452)	Dragging location is between two rows. Indicates a drop location above *row* and below *row* - 1.

These are a few examples of how to specify a drop location:

	Row index	Row-relative location
On row 2	2	NSBrowserDropOn (page 452)
Between rows 2 and 3	3	NSBrowserDropAbove (page 452)
Below the last row	[sender numberOfRows]	NSBrowserDropAbove (page 452)

	Row index	Row-relative location
All rows	-1	NSBrowserDropOn (page 452)

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- registerForDraggedTypes: (page 3201) (NSView)

Declared In
NSBrowser.h

browser:willDisplayCell:atRow:column:

Gives the delegate the opportunity to modify the specified cell at the given row and column location before the browser displays it.

```
- (void)browser:(NSBrowser *)sender
    willDisplayCell:(id)cell
    atRow:(NSInteger)row
    column:(NSInteger)column
```

Parameters

sender
The browser.

cell
The cell to be displayed.

row
The row index of the cell to be displayed.

column
The column index of the cell to be displayed.

Discussion
The delegate should set any state necessary for the correct display of the cell.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- browser:createRowsForColumn:inMatrix: (page 3586)
- browser:numberOfRowsInColumn: (page 3591)

Declared In
NSBrowser.h

browser:writeRowsWithIndexes:inColumn:toPasteboard:

Determines whether a drag operation can proceed. This method is required for a browser to be a drag source.

```
- (BOOL)browser:(NSBrowser *)browser
    writeRowsWithIndexes:(NSIndexSet *)rowIndexes
    inColumn:(NSInteger)column
    toPasteboard:(NSPasteboard *)pasteboard
```

Parameters

browser

The browser.

rowIndexes

The indexes of the rows the user is dragging.

column

The index of the column containing the dragged rows.

pasteboard

The pasteboard containing the content from the dragged rows.

Return Value

`YES` to allow the dragging operation to proceed (see discussion for further details); `NO` to disallow it.

Discussion

This method is called after a drag operation has been allowed to start (`browser:canDragRowsWithIndexes:inColumn:withEvent:` (page 3585) returns `YES`), but before it actually begins.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSBrowser.h`

browserColumnConfigurationDidChange:

Used by clients to implement their own column width persistence.

```
- (void)browserColumnConfigurationDidChange:(NSNotification *)notification
```

Parameters

notification

A notification named `NSBrowserColumnConfigurationDidChangeNotification` (page 453).

Discussion

This method applies only to browsers with resize type `NSBrowserUserColumnResizing` (page 452). It is invoked when the `setWidth:ofColumn:` (page 448) method of `NSBrowser` is used to change the width of any browser columns or when the user resizes any columns. If the user resizes more than one column, a single notification is posted when the user is finished resizing.

Availability

Available in Mac OS X v10.3 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `setWidth:ofColumn:` (page 448) (`NSBrowser`)

Declared In
`NSBrowser.h`

browserDidScroll:

Notifies the delegate when the browser has scrolled.

```
- (void)browserDidScroll:(NSBrowser *)sender
```

Parameters

sender
The browser sending the message.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSBrowser.h`

browserWillScroll:

Notifies the delegate when the browser will scroll.

```
- (void)browserWillScroll:(NSBrowser *)sender
```

Parameters

sender
The browser sending the message.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSBrowser.h`

rootItemForBrowser:

Asks the delegate to return the root item of the browser.

```
- (id)rootItemForBrowser:(NSBrowser *)browser
```

Parameters

browser
The browser.

Return Value
The browser's root item.

Discussion
By default, `nil` identifies the root item. This method can specify a different root item. To reload the previously set root item, call `loadColumnZero` (page 421), and `rootItemForBrowser:` will be called again.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSBrowser.h`

NSChangeSpelling Protocol Reference

Adopted by	NSText
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSSpellProtocol.h
Companion guide	Spell Checking

Overview

This protocol is implemented by objects in the responder chain that can correct a misspelled word. See the NSSpellChecker class description for more information.

Tasks

Changing Spellings

`- changeSpelling:` (page 3605) *required method*

Replaces the selected word in the receiver with a corrected version from the Spelling panel. (required)

Instance Methods

changeSpelling:

Replaces the selected word in the receiver with a corrected version from the Spelling panel. (required)

```
- (void)changeSpelling:(id)sender
```

Discussion

This message is sent by the `NSSpellChecker` to the object whose text is being checked. To get the corrected spelling, ask *sender* for the string value of its selected cell (visible to the user as the text field in the Spelling panel). This method should replace the selected portion of the text with the string that it gets from the NSSpellChecker.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSSpellProtocol.h`

NSCollectionViewDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSCollectionView.h
Related sample code	IconCollection

Overview

The `NSCollectionViewDelegate` protocol defines the optional methods implemented by delegates of `NSCollectionView` objects.

The `NSCollectionViewDelegate` provides support for both drag and drop, and pasteboard support to collection views.

Tasks

Drag and Drop Support

`- collectionView:canDragItemsAtIndexes:withEvent:` (page 3608)
Returns whether the collection view can attempt to initiate a drag for the given event and items.

`- collectionView:validateDrop:proposedIndex:dropOperation:` (page 3610)
Invoked to determine a valid drop target.

`- collectionView:acceptDrop:index:dropOperation:` (page 3608)
Invoked when the mouse is released over a collection view that previously allowed a drop.

`- collectionView:draggingImageForItemsAtIndexes:withEvent:offset:` (page 3609)
Sent to the delegate to allow creation of a custom image to represent collection view items during a drag operation.

`- collectionView:namesOfPromisedFilesDroppedAtDestination:forDraggedItemsAtIndexes:` (page 3610)
Invoked to return an array of filenames that the receiver promises to create.

Writing to the Pasteboard

`- collectionView:writeItemsAtIndexes:toPasteboard:` (page 3611)
Invoked after it has been determined that a drag should begin, but before the drag has been started.

Instance Methods

collectionView:acceptDrop:index:dropOperation:

Invoked when the mouse is released over a collection view that previously allowed a drop.

```
- (BOOL)collectionView:(NSCollectionView *)collectionView acceptDrop:(id <
    NSDraggingInfo >)draggingInfo index:(NSInteger)index
    dropOperation:(NSCollectionViewDropOperation)dropOperation
```

Parameters

collectionView
The collection view that send the message.

draggingInfo
An object that contains more information about this dragging operation.

index
The index of the proposed drop item.

dropOperation
The type of dragging operation.

Return Value
`YES` if the drop operation should be accepted, otherwise `NO`.

Discussion
This method is called when the mouse is released over a collection view that previously decied to allow a drop via the `collectionView:validateDrop:proposedIndex:dropOperation:` (page 3610) method. At this time, the delegate should incorporate the data from the dragging pasteboard and update the collection view's contents.

You must implement this method for your collection view to be a drag destination

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSCollectionView.h`

collectionView:canDragItemsAtIndexes:withEvent:

Returns whether the collection view can attempt to initiate a drag for the given event and items.

```
- (BOOL)collectionView:(NSCollectionView *)collectionView
    canDragItemsAtIndexes:(NSIndexSet *)indexes withEvent:(NSEvent *)event
```

Parameters

collectionView

The collection view that send the message.

indexes

The indexes of the proposed dragging items.

event

The mouse down event that initiated the drag.

Return Value

YES if the items can attempt to initiate a drag for the specified items, otherwise NO.

Discussion

If the delegate does not implement this method, the collection view will act as if it returned YES.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSCollectionView.h

collectionView:draggingImageForItemsAtIndexes:withEvent:offset:

Sent to the delegate to allow creation of a custom image to represent collection view items during a drag operation.

```
- (NSImage *)collectionView:(NSCollectionView *)collectionView
    draggingImageForItemsAtIndexes:(NSIndexSet *)indexes withEvent:(NSEvent *)event
     offset:(NSPointPointer)dragImageOffset
```

Parameters

collectionView

The collection view that send the message.

indexes

The indexes of the dragging items.

event

The mouse down event that initiated the drag.

dragImageOffset

An in/out parameter that will initially be set to NSZeroPoint. it can be modified to reposition the returned image. A *dragImageOffset* of NSZeroPoint will cause the image to be centered under the mouse.

Return Value

An image containing a rendering of the visible portions of the views for each item.

Discussion

If the delegate does not implement this method, the collection view with return an image using draggingImageForItemsAtIndexes:withEvent:offset: (page 645). You can safely invoked draggingImageForItemsAtIndexes:withEvent:offset: (page 645) on *collectionView* from within this method.

You do not need to implement this method for your collection view to be a drag source.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSCollectionView.h

collectionView:namesOfPromisedFilesDroppedAtDestination: forDraggedItemsAtIndexes:

Invoked to return an array of filenames that the receiver promises to create.

```
- (NSArray *)collectionView:(NSCollectionView *)collectionView
    namesOfPromisedFilesDroppedAtDestination:(NSURL *)dropURL
    forDraggedItemsAtIndexes:(NSIndexSet *)indexes
```

Parameters

collectionView
The collection view that send the message.

dropURL
The drop location where the files are created.

indexes
The indexes of the dragging items.

Return Value
An array of filenames (not full paths) for the created files that the receiver promises to create.

Discussion
The delegate can support file promise drags by adding NSFilesPromisePboardType to the pasteboard in `collectionView:writeItemsAtIndexes:toPasteboard:` (page 3611).

For more information on file promise dragging, see documentation for the `NSDraggingSource` protocol and `namesOfPromisedFilesDroppedAtDestination:` (page 3663).

You do not need to implement this delegate method for your collection view to be a drag source.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSCollectionView.h

collectionView:validateDrop:proposedIndex:dropOperation:

Invoked to determine a valid drop target.

```
- (NSDragOperation)collectionView:(NSCollectionView *)collectionView validateDrop:(id
    < NSDraggingInfo >)draggingInfo proposedIndex:(NSInteger *)proposedDropIndex
    dropOperation:(NSCollectionViewDropOperation *)proposedDropOperation
```

Parameters

collectionView
The collection view that send the message.

draggingInfo
An object containing details about this dragging operation.

proposedDropIndex
The proposed drop index. This parameter is passed by-reference and can be modified retarget the drop operation.

proposedDropOperation
The proposed drop operation. This parameter is passed by-reference and can be modified to change the drop operation.

Return Value
A value that indicates which dragging operation the data source will perform. It must return something other than `NSDragOperationNone` (page 3666) to accept the drop

Discussion
Based on the mouse position, the collection view will suggest a proposed index and drop operation. These values are in/out parameters and can be changed by the delegate to retarget the drop operation.

The collection view will propose `NSCollectionViewDropOn` when the dragging location is closer to the middle of the item than either of its edges. Otherwise, it will propose `NSCollectionViewDropBefore`. You may override this default behavior by changing *proposedDropOperation* or *proposedDropIndex*.

To receive drag messages, you must first send `registerForDraggedTypes:` (page 3201) to the collection view with the drag types you want to support.

You must implement this method for your collection view to be a drag destination.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSCollectionView.h`

collectionView:writeItemsAtIndexes:toPasteboard:

Invoked after it has been determined that a drag should begin, but before the drag has been started.

```
- (BOOL)collectionView:(NSCollectionView *)collectionView
    writeItemsAtIndexes:(NSIndexSet *)indexes toPasteboard:(NSPasteboard *)pasteboard
```

Parameters
collectionView
The collection view that send the message.

indexes
The indexes of the items to write to the pasteboard.

pasteboard
The pasteboard containing the content from the dragged items.

Return Value
`YES` to begin the drag, otherwise `NO`.

Discussion
To start the drag, you must first declare the pasteboard types that are supported by sending *pasteboard* a `declareTypes:owner:` (page 1893) method. You then place the data for the items at the specified indexes on *pasteboard*, and return `YES` from the method.

The drag image and other drag related information will be set up and provided by the view once this call returns `YES`.

You need to implement this method for your collection view to be a drag source.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSCollectionView.h`

NSColorPickingCustom Protocol Reference

Adopted by	NSColorPicker
Conforms to	NSColorPickingDefault
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSColorPicking.h
Companion guide	Color Programming Topics
Related sample code	RadiantColorPicker

Overview

Together with the `NSColorPickingDefault` protocol, `NSColorPickingCustom` provides a way to add color pickers—custom user interfaces for color selection—to an application's `NSColorPanel` instance. The `NSColorPickingDefault` protocol provides basic behavior for a color picker. The `NSColorPicker` class adopts the `NSColorPickingDefault` protocol.

Note: This protocol must be implemented by a custom picker, or an error will occur.

Tasks

Configuring Color Pickers

`- setColor:` (page 3615) *required method*
Adjusts the receiver to make the specified color the currently selected color. (required)

Getting Color Picker Information

`- currentMode` (page 3614) *required method*
Returns the receiver's current mode (or submode, if applicable). (required)

- supportsMode: (page 3615) *required method*
  Returns a Boolean value indicating whether or not the receiver supports the specified picking mode. (required)

Displaying Color Pickers

- provideNewView: (page 3614) *required method*
  Returns the view containing the receiver's user interface. (required)

Instance Methods

currentMode

Returns the receiver's current mode (or submode, if applicable). (required)

```
- (NSColorPanelMode)currentMode
```

Return Value
The current color picker mode. The returned value should be unique to your color picker. See this protocol description's list of the unique values for the standard color pickers used by the Application Kit.

Availability
Available in Mac OS X v10.0 and later.

See Also
- supportsMode: (page 3615)

Related Sample Code
RadiantColorPicker

Declared In
NSColorPicking.h

provideNewView:

Returns the view containing the receiver's user interface. (required)

```
- (NSView *)provideNewView:(BOOL)initialRequest
```

Parameters
initialRequest
YES only when this method is first invoked for your color picker. If *initialRequest* is YES, the method should perform any initialization required (such as lazily loading a nib file, initializing the view, or performing any other custom initialization required for your picker).

Return Value
The view containing the color picker's user interface. The NSView returned by this method should be set to automatically resize both its width and height.

Discussion
This message is sent to the color picker whenever the color panel attempts to display it. This may be when the panel is first presented, when the user switches pickers, or when the picker is switched through an API.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorPicking.h`

setColor:

Adjusts the receiver to make the specified color the currently selected color. (required)

```
- (void)setColor:(NSColor *)color
```

Parameters

color

The color to set as the currently selected color.

Discussion
This method is invoked on the current color picker each time `NSColorPanel`'s `setColor:` (page 733) method is invoked. If *color* is actually different from the color picker's color (as it would be if, for example, the user dragged a color into `NSColorPanel`'s color well), this method could be used to update the color picker's color to reflect the change.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorPicking.h`

supportsMode:

Returns a Boolean value indicating whether or not the receiver supports the specified picking mode. (required)

```
- (BOOL)supportsMode:(NSColorPanelMode)mode
```

Parameters

mode

The color picking mode.

Return Value
`YES` if the color picker supports the specified color picking mode; otherwise `NO`.

Discussion
This method is invoked when the `NSColorPanel` is first initialized: It is used to attempt to restore the user's previously selected mode. It is also invoked by `NSColorPanel`'s `setMode:` (page 734) method to find the color picker that supports a particular mode. See this protocol description's list of the unique mode values for the standard color pickers used by the Application Kit.

Availability
Available in Mac OS X v10.0 and later.

See Also
- currentMode (page 3614)

Declared In
NSColorPicking.h

NSColorPickingDefault Protocol Reference

Adopted by	NSColorPicker
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSColorPicking.h
Companion guide	Color Programming Topics

Overview

The `NSColorPickingDefault` protocol, together with the `NSColorPickingCustom` protocol, provides an interface for adding color pickers—custom user interfaces for color selection—to an application's `NSColorPanel` instance. The `NSColorPickingDefault` protocol provides basic behavior for a color picker. The `NSColorPickingCustom` protocol provides implementation-specific behavior.

Tasks

Creating Color Pickers

- `initWithPickerMask:colorPanel:` (page 3620)
  Initializes the receiver with a given color panel and its mode.

Configuring Color Pickers

- `setMode:` (page 3622)
  Specifies the receiver's mode.
- `insertNewButtonImage:in:` (page 3621)
  Sets the image of a given button cell.
- `provideNewButtonImage` (page 3621)
  Provides the image of the button used to select the receiver in the color panel.
- `minContentSize` (page 3621)
  Indicates the receiver's minimum content size.

- `buttonToolTip` (page 3619)
  Provides the toolbar button help tag.

Handling Events

- `alphaControlAddedOrRemoved:` (page 3618)
  Sent when the color panel's opacity controls have been hidden or displayed.
- `viewSizeChanged:` (page 3622)
  Tells the recever when the color panel's view size changes in a way that might affect the color picker.

Managing Color Lists

- `attachColorList:` (page 3618)
  Tells the receiver to attach the given color list, if it isn't already displaying the list.
- `detachColorList:` (page 3619)
  Tells the receiver to detach the given color list, unless the receiver isn't displaying the list.

Instance Methods

alphaControlAddedOrRemoved:

Sent when the color panel's opacity controls have been hidden or displayed.

```
- (void)alphaControlAddedOrRemoved:(id)sender
```

Parameters

sender
  The color panel sending the message.

Discussion
This method is invoked automatically when the opacity slider of the `NSColorPanel` is added or removed; you never invoke this method directly.

If the color picker has its own opacity controls, it should hide or display them, depending on whether the sender's `showsAlpha` (page 736) method returns `NO` or `YES`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSColorPicking.h`

attachColorList:

Tells the receiver to attach the given color list, if it isn't already displaying the list.

```
- (void)attachColorList:(NSColorList *)colorList
```

Parameters

colorList

The color list to display.

Discussion

You never invoke this method; it's invoked automatically by the `NSColorPanel` object when its `attachColorList:` (page 730) method is invoked. Because the `NSColorPanel` list mode manages `NSColorList` objects, this method need only be implemented by a custom color picker that manages `NSColorList` objects itself.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- detachColorList:` (page 3619)

Declared In

`NSColorPicking.h`

buttonToolTip

Provides the toolbar button help tag.

```
- (NSString *)buttonToolTip
```

Return Value

Help tag text.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSColorPicking.h`

detachColorList:

Tells the receiver to detach the given color list, unless the receiver isn't displaying the list.

```
- (void)detachColorList:(NSColorList *)colorList
```

Parameters

colorList

The color list to detach.

Discussion

You never invoke this method; it's invoked automatically by the `NSColorPanel` object when its `detachColorList:` (page 731) method is invoked. Because the `NSColorPanel` list mode manages `NSColorList` objects, this method need only be implemented by a custom color picker that manages `NSColorList` objects itself.

Availability

Available in Mac OS X v10.0 and later.

See Also
- attachColorList: (page 3618)

Declared In
NSColorPicking.h

initWithPickerMask:colorPanel:

Initializes the receiver with a given color panel and its mode.

```
- (id)initWithPickerMask:(NSUInteger)panelModes colorPanel:(NSColorPanel
    *)owningColorPanel
```

Parameters

panelModes

A mask indicating the various color picker modes supported by the color panel. This is determined by the argument to the NSColorPanel method setPickerMask: (page 728). If it has not been set, *panelModes* is NSColorPanelAllModesMask. If your color picker supports any additional modes, you should invoke the setPickerMask: (page 728) method when your application initializes to notify the NSColorPanel class. The standard mode constants are defined in "Choosing the Color Pickers in a Color Panel".

owningColorPanel

The color panel than owns the receiver.

Return Value
If your color picker responds to any of the modes represented in *panelModes*, it should perform its initialization and return an initialized color picker. Color pickers that do so have their buttons inserted in the color panel and continue to receive messages from the panel as the user manipulates it. If the color picker doesn't respond to any of the modes represented in *panelModes*, it should do nothing and return nil.

Discussion
This method is sent by the NSColorPanel to all implementors of the color-picking protocols when the application's color panel is first initialized. In order for your color picker to receive this message, it must have a bundle in your application's "ColorPickers" directory (described in "Color Picker Bundles").

This method should examine the mask and determine whether it supports any of the modes included there. You may also check the value in *mask* to enable or disable any subpickers or optional controls implemented by your color picker. Your color picker may also retain *owningColorPanel* in an instance variable for future communication with the color panel.

This method is provided to initialize your color picker; however, much of a color picker's initialization may be done lazily through the NSColorPickingCustom protocol's provideNewView: (page 3614) method.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ setPickerMask: (page 728) (NSColorPanel class)

Declared In
NSColorPicking.h

insertNewButtonImage:in:

Sets the image of a given button cell.

```
- (void)insertNewButtonImage:(NSImage *)newButtonImage in:(NSButtonCell *)buttonCell
```

Parameters

newButtonImage

The image to set for the button cell.

buttonCell

The `NSButtonCell` object that lets the user choose the picker from the color panel—the color picker's representation in the `NSMatrix` of the `NSColorPanel`.

Discussion
This method should perform application-specific manipulation of the image before it's inserted and displayed by the button cell.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- provideNewButtonImage` (page 3621)

Declared In
`NSColorPicking.h`

minContentSize

Indicates the receiver's minimum content size.

```
- (NSSize)minContentSize
```

Discussion
The receiver does not allow a size smaller than `minContentSize`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSColorPicking.h`

provideNewButtonImage

Provides the image of the button used to select the receiver in the color panel.

```
- (NSImage *)provideNewButtonImage
```

Return Value
The image for the mode button the user uses to select this picker in the color panel; that is, the color picker's representation in the `NSMatrix` of the `NSColorPanel`.

This image is the same one the color panel uses as an argument when sending the `insertNewButtonImage:in:` (page 3621) message.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSColorPicking.h

setMode:

Specifies the receiver's mode.

```
- (void)setMode:(NSColorPanelMode)mode
```

Parameters

mode

The color picker mode. The available modes are described in "Choosing the Color Pickers in a Color Panel".

Discussion
This method is invoked by the NSColorPanel method setMode: (page 734) method to ensure the color picker reflects the current mode. For example, invoke this method during color picker initialization to ensure that all color pickers are restored to the mode the user left them in the last time an NSColorPanel was used.

Most color pickers have only one mode and thus don't need to do any work in this method. An example of a color picker that uses this method is the slider picker, which can choose from one of several submodes depending on the value of *mode*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSColorPicking.h

viewSizeChanged:

Tells the recever when the color panel's view size changes in a way that might affect the color picker.

```
- (void)viewSizeChanged:(id)sender
```

Parameters

sender

The NSColorPanel that contains the color picker.

Discussion
Use this method to perform special preparation when resizing the color picker's view. Because this method is invoked only as appropriate, it's better to implement this method than to override the method superviewSizeChanged: for the NSView in which the color picker's user interface is contained.

Availability
Available in Mac OS X v10.0 and later.

See Also
- provideNewView: (page 3614) (NSColorPickingCustom protocol)

Declared In
`NSColorPicking.h`

NSComboBoxCellDataSource Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSComboBoxCell.h
Companion guide	Combo Box Programming Topics

Overview

The `NSComboBoxCellDataSource` protocol declares the methods that an `NSComboBoxCell` uses to access the contents of its data source object.

For more information, see *Combo Box Programming Topics*.

Tasks

Populating Combo Boxes

- `comboBoxCell:objectValueForItemAtIndex:` (page 3627)
  Returns the object that corresponds to the item at the given index in the combo box cell.
- `numberOfItemsInComboBoxCell:` (page 3627)
  Returns the number of items managed for the combo box cell by your data source object.

Entry Completion

- `comboBoxCell:completedString:` (page 3626)
  Returns the item from the combo box's pop-up list that matches the text entered by the user.
- `comboBoxCell:indexOfItemWithStringValue:` (page 3626)
  Invoked by an `NSComboBoxCell` object to synchronize the pop-up list's selected item with the text field's contents.

Instance Methods

comboBoxCell:completedString:

Returns the item from the combo box's pop-up list that matches the text entered by the user.

```
- (NSString *)comboBoxCell:(NSComboBoxCell *)aComboBoxCell completedString:(NSString
    *)uncompletedString
```

Parameters

aComboBoxCell

The combo box cell.

uncompletedString

The substring containing the text the user typed into the text field of the combo box cell.

Return Value

The completed string, from the items in the pop-up list, that matches the text entered by the user. Your implementation should return the first complete string that starts with *uncompletedString*.

Discussion

An `NSComboBoxCell` object uses this method to perform incremental—or "smart"—searches when the user types into the text field.

As the user types in the text field, the receiver uses this method to search for items from the pop-up list that start with what the user has typed. The receiver adds the new text to the end of the field and selects the new text, so when the user types another character, it replaces the new text.

If you don't implement this method, the receiver does not perform incremental searches.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSComboBoxCell.h`

comboBoxCell:indexOfItemWithStringValue:

Invoked by an `NSComboBoxCell` object to synchronize the pop-up list's selected item with the text field's contents.

```
- (NSUInteger)comboBoxCell:(NSComboBoxCell *)aComboBoxCell
    indexOfItemWithStringValue:(NSString *)aString
```

Parameters

aComboBoxCell

The combo box cell.

aString

The string to match. If `comboBoxCell:completedString:` (page 3626) is implemented, *aString* is the string returned by that method. Otherwise, *aString* is the text that the user has typed.

Return Value
The index for the pop-up list item matching *aString*, or NSNotFound if no item matches.

Discussion
If you don't implement this method, the receiver does not synchronize the pop-up list's selected item with the text field's contents.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSComboBoxCell.h

comboBoxCell:objectValueForItemAtIndex:

Returns the object that corresponds to the item at the given index in the combo box cell.

```
- (id)comboBoxCell:(NSComboBoxCell *)aComboBoxCell
    objectValueForItemAtIndex:(NSInteger)index
```

Parameters
aComboBoxCell
The combo box cell for which to return the item.
index
The index of the item to return.

Return Value
The object corresponding to the item at the specified index in the given combo box cell.

Discussion
An NSComboBoxCell object uses this method to populate the items displayed in its pop-up list.

Important: While this method is marked as @optional in the protocol, **you must implement this method if you are not providing the data for the combo box using using Cocoa bindings**.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSComboBoxCell.h

numberOfItemsInComboBoxCell:

Returns the number of items managed for the combo box cell by your data source object.

```
- (NSInteger)numberOfItemsInComboBoxCell:(NSComboBoxCell *)aComboBoxCell
```

Parameters

aComboBoxCell

The combo box cell for which your data source manages items.

Return Value

The number of items your data source object manages.

Discussion

An `NSComboBoxCell` object uses this method to determine how many items it should display in its pop-up list.

Important: While this method is marked as `@optional` in the protocol, **you must implement this method if you are not providing the data for the combo box using using Cocoa bindings**.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSComboBoxCell.h`

NSComboBoxDataSource Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSComboBox.h
Companion guide	Combo Box Programming Topics

Overview

The `NSComboBoxDataSource` informal protocol declares the methods that an `NSComboBox` object uses to access the contents of its data source object.

For more information, see *Combo Box Programming Topics*.

Tasks

Populating Combo Boxes

- `comboBox:objectValueForItemAtIndex:` (page 3631)
  Returns the object that corresponds to the item at the specified index in the combo box.
- `numberOfItemsInComboBox:` (page 3631)
  Returns the number of items that the data source manages for the combo box.

String Completion

- `comboBox:completedString:` (page 3630)
  Returns the first item from the pop-up list that starts with the text the user has typed.
- `comboBox:indexOfItemWithStringValue:` (page 3630)
  Returns the index of the combo box item matching the specified string.

Instance Methods

comboBox:completedString:

Returns the first item from the pop-up list that starts with the text the user has typed.

```
- (NSString *)comboBox:(NSComboBox *)aComboBox completedString:(NSString
    *)uncompletedString
```

Parameters

aComboBox

The combo box.

uncompletedString

The string to match against items in the combo box's pop-up list. This is text that the user has typed.

Return Value

The first complete string from the items in the combo box's pop-up list that starts with the string in *uncompletedString*.

Discussion

An `NSComboBox` object uses this method to perform incremental—or "smart"—searches when the user types into the text field. As the user types in the text field, the receiver uses this method to search for items from the pop-up list that start with what the user has typed. The receiver adds the new text to the end of the field and selects the new text, so when the user types another character, it replaces the new text.

This method is optional. If you don't implement it, the receiver does not perform incremental searches.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSComboBox.h`

comboBox:indexOfItemWithStringValue:

Returns the index of the combo box item matching the specified string.

```
- (NSUInteger)comboBox:(NSComboBox *)aComboBox indexOfItemWithStringValue:(NSString
    *)aString
```

Parameters

aComboBox

The combo box.

aString

The string to match against the items in the combo box. If the datasource implements`comboBox:completedString:` (page 3630), this is the string returned by that method. Otherwise, it is the text that the user has typed.

Return Value

The index for the item that matches the specified string, or `NSNotFound` if no item matches.

Discussion
An `NSComboBox` object uses this method to synchronize the pop-up list's selected item with the text field's contents. If you don't implement this method the receiver does not synchronize the pop-up list's selected item with the text field's contents.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSComboBox.h`

comboBox:objectValueForItemAtIndex:

Returns the object that corresponds to the item at the specified index in the combo box.

```
- (id)comboBox:(NSComboBox *)aComboBox objectValueForItemAtIndex:(NSInteger)index
```

Parameters

aComboBox
The combo box.

index
The index of the item to return.

Return Value
The object corresponding to the specified index number.

Discussion
An `NSComboBox` object uses this method to populate the items displayed in its pop-up list.

Important: While this method is marked as `@optional` in the protocol, you must implement this method if you are not providing the data for the combo box using Cocoa bindings.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSComboBox.h`

numberOfItemsInComboBox:

Returns the number of items that the data source manages for the combo box.

```
- (NSInteger)numberOfItemsInComboBox:(NSComboBox *)aComboBox
```

Parameters

aComboBox
The combo box.

Return Value
The number of items that the data source object manages for the specified combo box.

Discussion
An `NSComboBox` object uses this method to determine how many items it should display in its pop-up list.

Important: While this method is marked as `@optional` in the protocol, you must implement this method if you are not providing the data for the combo box using Cocoa bindings.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSComboBox.h`

NSComboBoxDelegate Protocol Reference

Conforms to	NSTextFieldDelegate
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSComboBox.h
Companion guide	Combo Box Programming Topics

Overview

The `NSComboBoxDelegate` protocol defines the optional methods implemented by delegates of `NSComboBox` objects.

Tasks

Manipulating the Selection

- `comboBoxSelectionDidChange:` (page 3633)
  Informs the delegate that the pop-up list selection has finished changing.
- `comboBoxSelectionIsChanging:` (page 3634)
  Informs the delegate that the pop-up list selection is changing.
- `comboBoxWillDismiss:` (page 3634)
  Informs the delegate that the pop-up list is about to be dismissed.
- `comboBoxWillPopUp:` (page 3634)
  Informs the delegate that the pop-up list is about to be displayed.

Instance Methods

comboBoxSelectionDidChange:

Informs the delegate that the pop-up list selection has finished changing.

```
- (void)comboBoxSelectionDidChange:(NSNotification *)notification
```

Parameters

notification

A notification named `NSComboBoxSelectionDidChangeNotification` (page 786).

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSComboBox.h`

comboBoxSelectionIsChanging:

Informs the delegate that the pop-up list selection is changing.

```
- (void)comboBoxSelectionIsChanging:(NSNotification *)notification
```

Parameters

notification

A notification named `NSComboBoxSelectionIsChangingNotification` (page 786).

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSComboBox.h`

comboBoxWillDismiss:

Informs the delegate that the pop-up list is about to be dismissed.

```
- (void)comboBoxWillDismiss:(NSNotification *)notification
```

Parameters

notification

A notification named `NSComboBoxWillDismissNotification` (page 786).

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSComboBox.h`

comboBoxWillPopUp:

Informs the delegate that the pop-up list is about to be displayed.

```
- (void)comboBoxWillPopUp:(NSNotification *)notification
```

Parameters

notification

A notification named `NSComboBoxWillPopUpNotification` (page 787).

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSComboBox.h`

NSControlTextEditingDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSControl.h
Companion guide	Control and Cell Programming Topics for Cocoa
Related sample code	CoreAnimationText

Overview

The `NSControlTextEditingDelegate` protocol defines the optional methods implemented by delegates of objects that are subclasses of `NSControl`.

Tasks

Validating a Control's Value

`- control:isValidObject:` (page 3639)
Invoked when the insertion point leaves a cell belonging to the specified control, but before the value of the cell's object is displayed.

`- control:didFailToValidatePartialString:errorDescription:` (page 3639)
Invoked when the formatter for the cell belonging to *`control`* (or selected cell) rejects a partial string a user is typing into the cell.

Responding to Text Formatting

`- control:didFailToFormatString:errorDescription:` (page 3638)
Invoked when the formatter for the cell belonging to the specified control cannot convert a string to an underlying object.

Responding to Text Editing

- `control:textShouldBeginEditing:` (page 3640)
  Invoked when the user tries to enter a character in a cell of a control that allows editing of text (such as a text field or form field).
- `control:textShouldEndEditing:` (page 3640)
  Invoked when the insertion point tries to leave a cell of the control that has been edited.

Working with Text Completion

- `control:textView:completions:forPartialWordRange:indexOfSelectedItem:` (page 3641)
  Invoked to allow you to control the list of proposed text completions generated by text fields and other controls.

Working with Key Bindings

- `control:textView:doCommandBySelector:` (page 3642)
  Invoked when users press keys with predefined bindings in a cell of the specified control.

Instance Methods

control:didFailToFormatString:errorDescription:

Invoked when the formatter for the cell belonging to the specified control cannot convert a string to an underlying object.

```
- (BOOL)control:(NSControl *)control didFailToFormatString:(NSString *)string
    errorDescription:(NSString *)error
```

Parameters

control
The control whose cell could not convert the string.

string
The string that could not be converted.

error
A localized, user-presentable string that explains why the conversion failed.

Return Value
`YES` if the value in the string parameter should be accepted as is; otherwise, `NO` if the value in the parameter should be rejected.

Discussion
Your implementation of this method should evaluate the error or query the user an appropriate value indicating whether the string should be accepted or rejected.

Availability
Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`getObjectValue:forString:errorDescription:` (`NSFormatter`)

Declared In
`NSControl.h`

control:didFailToValidatePartialString:errorDescription:

Invoked when the formatter for the cell belonging to *control* (or selected cell) rejects a partial string a user is typing into the cell.

```
- (void)control:(NSControl *)control didFailToValidatePartialString:(NSString
    *)string errorDescription:(NSString *)error
```

Parameters

control
The control whose cell rejected the string.

string
The string that includes the character that caused the rejection.

error
A localized, user-presentable string that explains why the string was rejected.

Discussion
You can implement this method to display a warning message or perform a similar action when the user enters improperly formatted text.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`isPartialStringValid:newEditingString:errorDescription:` (`NSFormatter`)

Declared In
`NSControl.h`

control:isValidObject:

Invoked when the insertion point leaves a cell belonging to the specified control, but before the value of the cell's object is displayed.

```
- (BOOL)control:(NSControl *)control isValidObject:(id)object
```

Parameters

control
The control whose object value needs to be validated.

object
The object value to validate.

Return Value
`YES` if you want to allow the control to display the specified value; otherwise, `NO` to reject the value and return the cursor to the control's cell.

Discussion
This method gives the delegate the opportunity to validate the contents of the control's cell (or selected cell). In validating, the delegate should check the value in the *`object`* parameter and determine if it falls within a permissible range, has required attributes, accords with a given context, and so on. Examples of objects subject to such evaluations are an `NSDate` object that should not represent a future date or a monetary amount (represented by an `NSNumber` object) that exceeds a predetermined limit.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSControl.h`

control:textShouldBeginEditing:

Invoked when the user tries to enter a character in a cell of a control that allows editing of text (such as a text field or form field).

```
- (BOOL)control:(NSControl *)control textShouldBeginEditing:(NSText *)fieldEditor
```

Parameters
`control`
The control whose content is about to be edited.

`fieldEditor`
The field editor of the control.

Return Value
`YES` if the control's field editor should be allowed to start editing the text; otherwise, `NO`.

Discussion
You can use this method to allow or disallow editing in a control. This message is sent by the control directly to its delegate object.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSControl.h`

control:textShouldEndEditing:

Invoked when the insertion point tries to leave a cell of the control that has been edited.

```
- (BOOL)control:(NSControl *)control textShouldEndEditing:(NSText *)fieldEditor
```

Parameters

control

The control for which editing is about to end.

fieldEditor

The field editor of the control. You can use this parameter to get the edited text.

Return Value

YES if the insertion point should be allowed to end the editing session; otherwise, NO.

Discussion

This message is sent only by controls that allow editing of text (such as a text field or a form field). This message is sent by the control directly to its delegate object.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSControl.h

control:textView:completions:forPartialWordRange:indexOfSelectedItem:

Invoked to allow you to control the list of proposed text completions generated by text fields and other controls.

```
- (NSArray *)control:(NSControl *)control textView:(NSTextView *)textView
    completions:(NSArray *)words forPartialWordRange:(NSRange)charRange
    indexOfSelectedItem:(NSInteger *)index
```

Parameters

control

The control whose cell initiated the message. If the control contains multiple cells, the one that initiated the message is usually the selected cell.

textView

The field editor of the control. You can use this parameter to get the typed text.

words

An array of NSString objects containing the potential completions. The completion strings are listed in their order of preference in the array.

charRange

The range of characters the user has already typed.

index

On input, an integer variable with the default value of 0. On output, you can set this value to an index in the returned array indicating which item should be selected initially. Set the value to -1 to indicate there should not be an initial selection.

Return Value

An array of NSString objects containing the list of completions to use in place of the array in the *words* parameter. The returned array should list the completions in their preferred order

Discussion
Each string you return should be a complete word that the user might be trying to type. The strings must be complete words rather than just the remainder of the word, in case completion requires some slight modification of what the user has already typed—for example, the addition of an accent, or a change in capitalization. You can also use this method to support abbreviations that complete into words that do not start with the characters of the abbreviation. The *index* argument allows you to return by reference an index specifying which of the completions should be selected initially.

The actual means of presentation of the potential completions is determined by the `complete:` method of `NSTextView`.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- complete:` (page 2888) (`NSTextView`)

Declared In
`NSControl.h`

control:textView:doCommandBySelector:

Invoked when users press keys with predefined bindings in a cell of the specified control.

```
- (BOOL)control:(NSControl *)control textView:(NSTextView *)textView
    doCommandBySelector:(SEL)command
```

Parameters

control
The control whose cell initiated the message. If the control contains multiple cells, the one that initiated the message is usually the selected cell.

textView
The field editor of the control.

command
The selector that was associated with the binding.

Return Value
`YES` if the delegate object handles the key binding; otherwise, `NO`.

Discussion
These bindings are usually implemented as methods (*command*) defined in the `NSResponder` class; examples of such key bindings are arrow keys (for directional movement) and the Escape key (for name completion). By implementing this method, the delegate can override the default implementation of *command* and supply its own behavior.

For example, the default method for completing partially typed pathnames or symbols (usually when users press the Escape key) is `complete:` (page 2147). The default implementation of the `complete:` method (in `NSResponder`) does nothing. The delegate could evaluate the method in the *command* parameter and, if it's `complete:`, get the current string from the *textView* parameter and then expand it, or display a list of potential completions, or do whatever else is appropriate.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSControl.h`

NSDatePickerCellDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSDatePickerCell.h
Related sample code	DatePicker

Overview

The `NSDatePickerCellDelegate` protocol defines the optional methods implemented by delegates of `NSDatePickerCell` objects.

Tasks

Content Validation

`- datePickerCell:validateProposedDateValue:timeInterval:` (page 3645)
The delegate receives this message each time the user attempts to change the receiver's value, allowing the delegate the opportunity to override the change.

Instance Methods

datePickerCell:validateProposedDateValue:timeInterval:

The delegate receives this message each time the user attempts to change the receiver's value, allowing the delegate the opportunity to override the change.

```
- (void)datePickerCell:(NSDatePickerCell *)aDatePickerCell
    validateProposedDateValue:(NSDate **)proposedDateValue
    timeInterval:(NSTimeInterval *)proposedTimeInterval
```

Parameters

aDatePickerCell

The cell cell that sent the message.

proposedDateValue

On input, contains the proposed new date. The delegate may change this value before returning.

proposedTimeInterval

On input, contains the proposed new time interval. The delegate may change this value before returning.

Discussion

When returning a new *proposedDateValue*, the `NSDate` instance should be autoreleased, and the *proposedDateValue* should not be released by the delegate.

The *proposedDateValue* and *proposedTimeInterval* are guaranteed to lie between the dates returned by `minDate` (page 899) and `maxDate` (page 898). If you modify these values, you should ensure that the new values are within the appropriate range.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSDatePickerCell.h`

NSDictionaryControllerKeyValuePair Protocol Reference

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSDictionaryController.h
Availability	Available in Mac OS X v10.5 and later.
Companion guide	Cocoa Bindings Programming Topics

Overview

`NSDictionaryControllerKeyValuePair` is an informal protocol that is implemented by objects returned by the `NSDictionaryController` method `arrangedObjects`. See *NSDictionaryController Class Reference* for more information.

Tasks

Localizing the Display Key

- `setLocalizedKey:` (page 3649)
  Sets the localized key name for the receiver.
- `localizedKey` (page 3648)
  Returns the receiver's localized key name.

Key-Value Pair Settings

- `setKey:` (page 3649)
  Sets the key name for the receiver.
- `key` (page 3648)
  Returns the receiver's key name.
- `setValue:` (page 3649)
  Sets the receiver's value.
- `value` (page 3650)
  Returns the receiver's value.
- `isExplicitlyIncluded` (page 3648)
  Specifies whether the receiver's key name is an included key.

Instance Methods

isExplicitlyIncluded

Specifies whether the receiver's key name is an included key.

```
- (BOOL)isExplicitlyIncluded
```

Return Value
`YES` if the key name is specified as an included key, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSDictionaryController.h`

key

Returns the receiver's key name.

```
- (NSString *)key
```

Return Value
The key name.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setKey:` (page 3649)

Related Sample Code
EnhancedAudioBurn
Quartz EB
SimpleStickies
STUCAuthoringDeviceCocoaSample
WhackedTV

Declared In
`NSDictionaryController.h`

localizedKey

Returns the receiver's localized key name.

```
- (NSString *)localizedKey
```

Return Value
The localized key name.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setLocalizedKey:` (page 3649)

Declared In
`NSDictionaryController.h`

setKey:

Sets the key name for the receiver.

```
- (void)setKey:(NSString *)key
```

Parameters

key
The key name.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `key` (page 3648)

Declared In
`NSDictionaryController.h`

setLocalizedKey:

Sets the localized key name for the receiver.

```
- (void)setLocalizedKey:(NSString *)localizedKey
```

Parameters

localizedKey
The localized name of the receivers key.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `localizedKey` (page 3648)

Declared In
`NSDictionaryController.h`

setValue:

Sets the receiver's value.

```
- (void)setValue:(id)value
```

Parameters

value

An object.

Availability
Available in Mac OS X v10.5 and later.

See Also
- value (page 3650)

Declared In
NSDictionaryController.h

value

Returns the receiver's value.

```
- (id)value
```

Return Value
The receiver's value object.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setValue: (page 3649)

Related Sample Code
CocoaHTTPServer
Quartz Composer QCTV
SetMouseAcclSample
SillyFrequencyLevels
WebKitPluginWithJavaScript

Declared In
NSDictionaryController.h

NSDockTilePlugIn Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSDockTile.h
Companion guide	Dock Tile Programming Guide

Overview

The `NSDockTilePlugIn` protocol defines the methods implemented by plug-ins that allow an application's Dock tile to be customized while the application is not running.

Customizing an application's Dock tile when the application itself is not running requires that you write a plug-in. The plug-in's principal class must implement the `NSDockTilePlugIn` protocol.

The name of the plugin is indicated by a `NSDockTilePlugIn` key in the application's Info.plist file.

The plugin is loaded in a system process at login time or when the application tile is added to the Dock. When the plugin is loaded, the principal class' implementation of `setDockTile:` (page 3652) is invoked, passing an `NSDockTile` for the plug-in to customize. If the principal class implements `dockMenu` (page 3652) it is invoked whenever the user causes the application's dock menu to be shown. When the dock tile is no longer valid (for example,. the application has been removed from the dock) -`setDockTile:` (page 3652) is invoked with `nil`.

Tasks

Setting the Dock Tile

- `setDockTile:` (page 3652) *required method*

  Invoked when the plug-in is first loaded and when the application is removed from the Dock. (required)

Getting the Dock Tile Menu

Instance Methods

dockMenu

Invoked when the user causes the application's dock menu to be shown.

```
- (NSMenu*)dockMenu
```

Return Value
The menu the dock tile displays.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSDockTile.h

setDockTile:

Invoked when the plug-in is first loaded and when the application is removed from the Dock. (required)

```
- (void)setDockTile:(NSDockTile*)dockTile
```

Parameters

dockTile
The dock tile associated with the application, or nil if the application has been removed from the Dock.

Discussion
The plugin is loaded in a system process at login time or when the application tile is added to the Dock.

The principal class of the plug-in must implement the NSDockTilePlugIn protocol.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSDockTile.h

NSDraggingDestination Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSDragging.h
Companion guide	Drag and Drop Programming Topics for Cocoa

Overview

The `NSDraggingDestination` informal protocol declares methods that the destination object (or recipient) of a dragged image must implement. The destination automatically receives `NSDraggingDestination` messages for pasteboard data types it has registered for as an image enters, moves around inside, and then exits or is released within the destination's boundaries.

Tasks

Managing a Dragging Session Before an Image Is Released

- `draggingEntered:` (page 3655)
  Invoked when the dragged image enters destination bounds or frame; delegate returns dragging operation to perform.
- `wantsPeriodicDraggingUpdates` (page 3658)
  Asks the destination object whether it wants to receive periodic `draggingUpdated:` (page 3656) messages.
- `draggingUpdated:` (page 3656)
  Invoked periodically as the image is held within the destination area, allowing modification of the dragging operation or mouse-pointer position.
- `draggingEnded:` (page 3654)
  Implement this method to be notified when a drag operation ends in some other destination.
- `draggingExited:` (page 3655)
  Invoked when the dragged image exits the destination's bounds rectangle (in the case of a view object) or its frame rectangle (in the case of a window object).

Managing a Dragging Session After an Image Is Released

- `prepareForDragOperation:` (page 3657)
  Invoked when the image is released, allowing the receiver to agree to or refuse drag operation.

- `performDragOperation:` (page 3657)
  Invoked after the released image has been removed from the screen, signaling the receiver to import the pasteboard data.
- `concludeDragOperation:` (page 3654)
  Invoked when the dragging operation is complete, signaling the receiver to perform any necessary clean-up.

Instance Methods

concludeDragOperation:

Invoked when the dragging operation is complete, signaling the receiver to perform any necessary clean-up.

```
- (void)concludeDragOperation:(id < NSDraggingInfo >)sender
```

Parameters

sender
The object sending the message; use it to get details about the dragging operation.

Discussion
For this method to be invoked, the previous `performDragOperation:` (page 3657) must have returned `YES`. The destination implements this method to perform any tidying up that it needs to do, such as updating its visual representation now that it has incorporated the dragged data. This message is the last message sent from *sender* to the destination during a dragging session.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDragging.h`

draggingEnded:

Implement this method to be notified when a drag operation ends in some other destination.

```
- (void)draggingEnded:(id < NSDraggingInfo >)sender
```

Parameters

sender
The object sending the message; use it to get details about the dragging operation.

Discussion
This method might be used by a destination doing auto-expansion in order to collapse any auto-expands.

Note: While this method has been implemented since Mac OS X V 10.0, it was non-functional until Mac OS X v10.5.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDragging.h

draggingEntered:

Invoked when the dragged image enters destination bounds or frame; delegate returns dragging operation to perform.

```
- (NSDragOperation)draggingEntered:(id < NSDraggingInfo >)sender
```

Parameters

sender

The object sending the message; use it to get details about the dragging operation.

Return Value
One (and only one) of the dragging operation constants described in `Dragging operations` (page 3664) in the `NSDraggingInfo` reference. The default return value (if this method is not implemented by the destination) is the value returned by the previous `draggingEntered:` (page 3655) message.

Discussion
Invoked when a dragged image enters the destination but only if the destination has registered for the pasteboard data type involved in the drag operation. Specifically, this method is invoked when the mouse pointer enters the destination's bounds rectangle (if it is a view object) or its frame rectangle (if it is a window object).

This method must return a value that indicates which dragging operation the destination will perform when the image is released. In deciding which dragging operation to return, the method should evaluate the overlap between both the dragging operations allowed by the source (obtained from *sender* with the `draggingSourceOperationMask` (page 3663) method) and the dragging operations and pasteboard data types the destination itself supports.

If none of the operations is appropriate, this method should return `NSDragOperationNone` (this is the default response if the method is not implemented by the destination). A destination will still receive `draggingUpdated:` (page 3656) and `draggingExited:` (page 3655) even if `NSDragOperationNone` is returned by this method.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `draggingUpdated:` (page 3656)
- `draggingExited:` (page 3655)
- `prepareForDragOperation:` (page 3657)

Declared In
NSDragging.h

draggingExited:

Invoked when the dragged image exits the destination's bounds rectangle (in the case of a view object) or its frame rectangle (in the case of a window object).

```
- (void)draggingExited:(id < NSDraggingInfo >)sender
```

Parameters

sender

The object sending the message; use it to get details about the dragging operation.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDragging.h

draggingUpdated:

Invoked periodically as the image is held within the destination area, allowing modification of the dragging operation or mouse-pointer position.

```
- (NSDragOperation)draggingUpdated:(id < NSDraggingInfo >)sender
```

Parameters

sender

The object sending the message; use it to get details about the dragging operation.

Return Value

One (and only one) of the dragging operation constants described in `Dragging operations` (page 3664) in the `NSDraggingInfo` reference. The default return value (if this method is not implemented by the destination) is the value returned by the previous `draggingEntered:` (page 3655) message.

Discussion

For this to be invoked, the destination must have registered for the pasteboard data type involved in the drag operation. The messages continue until the image is either released or dragged out of the window or view.

This method provides the destination with an opportunity to modify the dragging operation depending on the position of the mouse pointer inside of the destination view or window object. For example, you may have several graphics or areas of text contained within the same view and wish to tailor the dragging operation, or to ignore the drag event completely, depending upon which object is underneath the mouse pointer at the time when the user releases the dragged image and the `performDragOperation:` (page 3657) method is invoked.

You typically examine the contents of the pasteboard in the `draggingEntered:` (page 3655) method, where this examination is performed only once, rather than in the `draggingUpdated:` (page 3656) method, which is invoked multiple times.

Only one destination at a time receives a sequence of `draggingUpdated:` (page 3656) messages. If the mouse pointer is within the bounds of two overlapping views that are both valid destinations, the uppermost view receives these messages until the image is either released or dragged out.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `draggingExited:` (page 3655)
- `prepareForDragOperation:` (page 3657)

Declared In
NSDragging.h

performDragOperation:

Invoked after the released image has been removed from the screen, signaling the receiver to import the pasteboard data.

```
- (BOOL)performDragOperation:(id < NSDraggingInfo >)sender
```

Parameters

sender
The object sending the message; use it to get details about the dragging operation.

Return Value
If the destination accepts the data, it returns YES; otherwise it returns NO. The default is to return NO.

Discussion
For this method to be invoked, the previous prepareForDragOperation: (page 3657) message must have returned YES. The destination should implement this method to do the real work of importing the pasteboard data represented by the image.

Availability
Available in Mac OS X v10.0 and later.

See Also
- concludeDragOperation: (page 3654)

Declared In
NSDragging.h

prepareForDragOperation:

Invoked when the image is released, allowing the receiver to agree to or refuse drag operation.

```
- (BOOL)prepareForDragOperation:(id < NSDraggingInfo >)sender
```

Parameters

sender
The object sending the message; use it to get details about the dragging operation.

Return Value
YES if the receiver agrees to perform the drag operation and NO if not.

Discussion
This method is invoked only if the most recent draggingEntered: (page 3655) or draggingUpdated: (page 3656) message returned an acceptable drag-operation value.

Availability
Available in Mac OS X v10.0 and later.

See Also
- performDragOperation: (page 3657)

Declared In
NSDragging.h

wantsPeriodicDraggingUpdates

Asks the destination object whether it wants to receive periodic `draggingUpdated:` (page 3656) messages.

```
- (BOOL)wantsPeriodicDraggingUpdates
```

Return Value
`YES` if the destination wants to receive periodic `draggingUpdated:` (page 3656) messages, `NO` otherwise.

Discussion
If the destination returns `NO`, these messages are sent only when the mouse moves or a modifier flag changes. Otherwise the destination gets the default behavior, where it receives periodic dragging-updated messages even if nothing changes.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
GeekGameBoard

Declared In
NSDragging.h

NSDraggingInfo Protocol Reference

Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSDragging.h
Companion guide	Drag and Drop Programming Topics for Cocoa
Related sample code	CompositeLab DemoMonkey QTKitMovieShuffler Sketch-112 With and Without Bindings

Overview

The `NSDraggingInfo` protocol declares methods that supply information about a dragging session. `NSDraggingInfo` methods are designed to be invoked from within a class's implementation of `NSDraggingDestination` informal protocol methods. The Application Kit automatically passes an object that conforms to the `NSDraggingInfo` protocol as the argument to each of the methods defined by `NSDraggingDestination`. `NSDraggingInfo` messages should be sent to this object; you never need to create a class that implements the `NSDraggingInfo` protocol.

Tasks

Obtaining Information About the Dragging Session

- `draggingPasteboard` (page 3662)
  Returns the pasteboard object that holds the data being dragged.
- `draggingSequenceNumber` (page 3662)
  Returns a number that uniquely identifies the dragging session.
- `draggingSource` (page 3662)
  Returns the source, or owner, of the dragged data.
- `draggingSourceOperationMask` (page 3663)
  Returns the dragging operation mask of the dragging source.

- draggingLocation (page 3661)
  Returns the current location of the mouse pointer in the base coordinate system of the destination object's window.
- draggingDestinationWindow (page 3661)
  Returns the destination window for the dragging operation.
- namesOfPromisedFilesDroppedAtDestination: (page 3663)
  Sets the drop location for promised files and returns the names of the files that the receiver promises to create there.

Getting Image Information

- draggedImage (page 3660)
  Returns the image being dragged.
- draggedImageLocation (page 3661)
  Returns the current location of the dragged image's origin.

Sliding the Image

- slideDraggedImageTo: (page 3664)
  Slides the image to a specified location.

Instance Methods

draggedImage

Returns the image being dragged.

```
- (NSImage *)draggedImage
```

Return Value
The image being dragged.

Discussion
This image object visually represents the data put on the pasteboard during the drag operation; however, it is the pasteboard data and not this image that is ultimately utilized in the dragging operation.

Availability
Available in Mac OS X v10.0 and later.

See Also
- draggedImageLocation (page 3661)

Declared In
NSDragging.h

draggedImageLocation

Returns the current location of the dragged image's origin.

```
- (NSPoint)draggedImageLocation
```

Return Value
The dragged image's origin, in the base coordinate system of the destination object's window.

Discussion
The image moves along with the mouse pointer (the position of which is given by `draggingLocation` (page 3661)) but may be positioned at some offset.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- draggedImage` (page 3660)

Declared In
`NSDragging.h`

draggingDestinationWindow

Returns the destination window for the dragging operation.

```
- (NSWindow *)draggingDestinationWindow
```

Return Value
The destination window for the dragging operation.

Discussion
Either this window is the destination itself, or it contains the view object that is the destination.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSDragging.h`

draggingLocation

Returns the current location of the mouse pointer in the base coordinate system of the destination object's window.

```
- (NSPoint)draggingLocation
```

Return Value
The current location of the mouse pointer in the base coordinate system of the destination object's window.

Availability
Available in Mac OS X v10.0 and later.

See Also
- draggedImageLocation (page 3661)

Declared In
NSDragging.h

draggingPasteboard

Returns the pasteboard object that holds the data being dragged.

```
- (NSPasteboard *)draggingPasteboard
```

Return Value
The pasteboard object that holds the data being dragged.

Discussion
The dragging operation that is ultimately performed utilizes this pasteboard data and not the image returned by the draggedImage (page 3660) method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDragging.h

draggingSequenceNumber

Returns a number that uniquely identifies the dragging session.

```
- (NSInteger)draggingSequenceNumber
```

Return Value
A number that uniquely identifies the dragging session.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDragging.h

draggingSource

Returns the source, or owner, of the dragged data.

```
- (id)draggingSource
```

Return Value
The source, or owner, of the dragged data.

Discussion
This method returns nil if the source is not in the same application as the destination. The dragging source implements methods from the NSDraggingSource informal protocol.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDragging.h

draggingSourceOperationMask

Returns the dragging operation mask of the dragging source.

```
- (NSDragOperation)draggingSourceOperationMask
```

Return Value
The dragging operation mask, which is declared by the dragging source through its `draggingSourceOperationMaskForLocal:` (page 3670) method. If the source does not permit any dragging operations, this method should return `NSDragOperationNone`.

Discussion
If the source permits dragging operations, the elements in the mask are one or more of the constants described in "Obtaining Information About the Dragging Session" (page 3659), combined using the C bitwise OR operator.

If the user is holding down a modifier key during the dragging session and the source does not prohibit modifier keys from affecting the drag operation (through its `ignoreModifierKeysWhileDragging` (page 3670) method), then the operating system combines the dragging operation value that corresponds to the modifier key (see the descriptions below) with the source's mask using the C bitwise AND operator.

The modifier keys are associated with the dragging operation options shown below:

Modifier Key	Dragging Operation
Control	`NSDragOperationLink`
Option	`NSDragOperationCopy`
Command	`NSDragOperationGeneric`

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDragging.h

namesOfPromisedFilesDroppedAtDestination:

Sets the drop location for promised files and returns the names of the files that the receiver promises to create there.

```
- (NSArray *)namesOfPromisedFilesDroppedAtDestination:(NSURL *)dropDestination
```

Parameters

dropDestination

A URL object specifying the drop location for promised files.

Return Value

An array of file names, which are not full paths.

Discussion

Drag destinations should invoke this method within their `performDragOperation:` (page 3657) method. The source may or may not have created the files by the time this method returns.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSDragging.h`

slideDraggedImageTo:

Slides the image to a specified location.

```
- (void)slideDraggedImageTo:(NSPoint)aPoint
```

Parameters

aPoint

A point that specifies a location in the screen coordinate system.

Discussion

This method can be used to adjust the location to which the dragged image will slide back if the drag is rejected.

It should only be invoked from within the destination's implementation of `prepareForDragOperation:` (page 3657), and will only have effect if the destination rejects the drag.

This method is invoked after the user has released the image but before it is removed from the screen.

Special Considerations

This method has been available since Mac OS X v 10.0, however it was not implemented until Mac OS X v 10.5. Previous to that version, it did nothing.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDragging.h`

Constants

Dragging operations

These constants are used by `draggingSourceOperationMask` (page 3663).

```
enum {
   NSDragOperationNone    = 0,
   NSDragOperationCopy    = 1,
   NSDragOperationLink    = 2,
   NSDragOperationGeneric = 4,
   NSDragOperationPrivate = 8,
   NSDragOperationAll_Obsolete  = 15,
   NSDragOperationMove    = 16,
   NSDragOperationDelete  = 32,
   NSDragOperationEvery   = NSUIntegerMax
};
typedef NSUInteger NSDragOperation;
```

Constants

`NSDragOperationCopy`
The data represented by the image can be copied.
Available in Mac OS X v10.0 and later.
Declared in `NSDragging.h`.

`NSDragOperationLink`
The data can be shared.
Available in Mac OS X v10.0 and later.
Declared in `NSDragging.h`.

`NSDragOperationGeneric`
The operation can be defined by the destination.
Available in Mac OS X v10.0 and later.
Declared in `NSDragging.h`.

`NSDragOperationPrivate`
The operation is negotiated privately between the source and the destination.
Available in Mac OS X v10.0 and later.
Declared in `NSDragging.h`.

`NSDragOperationAll_Obsolete`
The `NSDragOperationAll` constant is deprecated. Use `NSDragOperationEvery` (page 3665) instead.
Available in Mac OS X v10.0 and later.
Declared in `NSDragging.h`.

`NSDragOperationMove`
The data can be moved.
Available in Mac OS X v10.0 and later.
Declared in `NSDragging.h`.

`NSDragOperationDelete`
The data can be deleted.
Available in Mac OS X v10.0 and later.
Declared in `NSDragging.h`.

`NSDragOperationEvery`
All of the above.
Available in Mac OS X v10.0 and later.
Declared in `NSDragging.h`.

`NSDragOperationNone`
No drag operations are allowed.

Available in Mac OS X v10.0 and later.

Declared in `NSDragging.h`.

NSDragOperationAll Deprecation

The NSDragOperationAll constant has been deprecated. Use `NSDragOperationEvery` (page 3665) instead.

```
#define NSDragOperationAll NSDragOperationAll_Obsolete
```

Constants

`NSDragOperationAll`
Use `NSDragOperationEvery` (page 3665) instead.

Available in Mac OS X v10.0 and later.

Declared in `NSDragging.h`.

NSDraggingSource Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSDragging.h
Companion guide	Drag and Drop Programming Topics for Cocoa

Overview

The `NSDraggingSource` informal protocol declares methods that are implemented by the source object in a dragging session. The dragging source is specified as an argument to the `dragImage:at:offset:event:pasteboard:source:slideBack:` (page 3168) message, sent to a window or view object to initiate the dragging session.

The `NSDraggingSource` methods are invoked only if the dragging source implements them. All methods are invoked automatically during a dragging session—you never send an `NSDraggingSource` message directly to an object.

Tasks

Specifying Dragging Options

- `draggingSourceOperationMaskForLocal:` (page 3670)
  Returns an integer bit mask indicating the types of dragging operations the source object will allow to be performed on the dragged image's data.
- `ignoreModifierKeysWhileDragging` (page 3670)
  Sets whether the use of modifier keys should have an effect on the type of operation performed.

Responding to Messages During a Dragging Session

- `draggedImage:beganAt:` (page 3668)
  Gives the source object an opportunity to respond to the initiation of a dragging session.
- `draggedImage:movedTo:` (page 3669)
  Informs the dragging source when a dragged image moves to a new screen coordinate.
- `draggedImage:endedAt:operation:` (page 3669)
  Invoked after the dragging destination has been given a chance to operate on the data represented by the image,

- `namesOfPromisedFilesDroppedAtDestination:` (page 3671)
  Returns the names of the files that the receiver promises to create at a specified location.
- `draggedImage:endedAt:deposited:` (page 3668) *required method* Deprecated in Mac OS X v10.1
  Invoked after the dragging destination has been given a chance to operate on the data represented by the image. (required) (Deprecated. Use `draggedImage:endedAt:operation:` (page 3669) instead.)

Instance Methods

draggedImage:beganAt:

Gives the source object an opportunity to respond to the initiation of a dragging session.

```
- (void)draggedImage:(NSImage *)anImage beganAt:(NSPoint)aPoint
```

Parameters

anImage
The image of the dragged item.

aPoint
The origin of the image in screen coordinates.

Discussion
This method is invoked when *anImage* is displayed but before it starts following the mouse. For example, you might choose to have the source give a visual indication to the user that data is being dragged from the source.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `convertScreenToBase:` (page 3314) (NSWindow)
- `convertBaseToScreen:` (page 3313) (NSWindow)
- `convertPoint:fromView:` (page 3155) (NSView)
- `convertPoint:toView:` (page 3155) (NSView)

Declared In
`NSDragging.h`

draggedImage:endedAt:deposited:

Invoked after the dragging destination has been given a chance to operate on the data represented by the image. (required) (Deprecated in Mac OS X v10.1. Use `draggedImage:endedAt:operation:` (page 3669) instead.)

```
- (void)draggedImage:(NSImage *)anImage endedAt:(NSPoint)aPoint deposited:(BOOL)flag
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.1.

Declared In
NSDragging.h

draggedImage:endedAt:operation:

Invoked after the dragging destination has been given a chance to operate on the data represented by the image,

```
- (void)draggedImage:(NSImage *)anImage endedAt:(NSPoint)aPoint
    operation:(NSDragOperation)operation
```

Parameters

anImage
The dragged image.

aPoint
The point locating the image's origin in the screen coordinate system when it was released.

operation
An integer constant that indicates the operation performed by the destination.

Discussion
This method is invoked after the dragged image (*anImage*) has been released and the dragging destination has been given a chance to operate on the data it represents. This method provides the source object with an opportunity to respond to either a successful or a failed dragging session. For example, if you are moving data from one location to another, you could use this method to make the source data disappear from its previous location, if the dragging session is successful, or reset itself to its previous state, in the event of a failure.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDragging.h

draggedImage:movedTo:

Informs the dragging source when a dragged image moves to a new screen coordinate.

```
- (void)draggedImage:(NSImage *)draggedImage movedTo:(NSPoint)screenPoint
```

Parameters

draggedImage
The dragged image.

screenPoint
The point specifying the new location of the image in screen coordinates.

Discussion
This message is similar to the dragging destination being sent `draggingUpdated:` (page 3656) messages.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDragging.h

draggingSourceOperationMaskForLocal:

Returns an integer bit mask indicating the types of dragging operations the source object will allow to be performed on the dragged image's data.

```
- (NSDragOperation)draggingSourceOperationMaskForLocal:(BOOL)isLocal
```

Parameters

isLocal

YES indicates that the candidate destination object (the window or view over which the dragged image is currently poised) is in the same application as the source, while a NO value indicates that the destination object is in a different application.

Return Value
A mask, created by combining the dragging operations listed in the NSDragOperation (page 3664) section of NSDraggingInfo protocol reference using the C bitwise OR operator.If the source does not permit any dragging operations, it should return NSDragOperationNone (page 3666).

Discussion
If not implemented, the default value is NSDragOperationCopy (page 3665) | NSDragOperationLink (page 3665) | NSDragOperationGeneric (page 3665) | NSDragOperationPrivate (page 3665).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDragging.h

ignoreModifierKeysWhileDragging

Sets whether the use of modifier keys should have an effect on the type of operation performed.

```
- (BOOL)ignoreModifierKeysWhileDragging
```

Return Value
Return NO, if the user can tailor the drag operation by holding down a modifier key during the drag.

Discussion
If this method is not implemented the default behavior is equivalent to returning NO

The dragging option that corresponds to the modifier key is combined with the source's mask (as set with the draggingSourceOperationMaskForLocal: (page 3670) method) using the C bitwise OR operator. See the description for the draggingSourceOperationMask (page 3663) method in the NSDraggingInfo protocol specification for more information about dragging masks and modifier keys.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSDragging.h

namesOfPromisedFilesDroppedAtDestination:

Returns the names of the files that the receiver promises to create at a specified location.

```
- (NSArray *)namesOfPromisedFilesDroppedAtDestination:(NSURL *)dropDestination
```

Parameters

dropDestination

A URL object that identifies the location at which the promised files will be created.

Return Value

An array of the names of files (not full paths) that the receiver promises to create at `dropDestination`.

Discussion

This method is invoked when the drop has been accepted by the destination and the destination, in the case of another Cocoa application, invokes the NSDraggingInfo method `namesOfPromisedFilesDroppedAtDestination:` (page 3663). For long operations, you can cache *dropDestination* and defer the creation of the files until the `draggedImage:endedAt:operation:` (page 3669) method to avoid blocking the destination application.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSDragging.h`

NSDrawerDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSDrawer.h
Companion guide	Drawers
Related sample code	DrawerMadness

Overview

The `NSDrawerDelegate` protocol defines the messages sent to delegates of `NSDrawer`. All of the methods of this protocol are optional.

Tasks

Opening and Closing Drawers

- `drawerShouldOpen:` (page 3675)
  Asks the delegate if the specified drawer should open.
- `drawerWillOpen:` (page 3676)
  Notifies the delegate that the drawer will open.
- `drawerDidOpen:` (page 3674)
  Notifies the delegate that the drawer has opened.
- `drawerShouldClose:` (page 3674)
  Asks the delegate if the specified drawer should close.
- `drawerWillClose:` (page 3675)
  Notifies the delegate the the drawer will close.
- `drawerDidClose:` (page 3674)
  Notifies the delegate that the drawer has closed.

Managing Drawer Size

`- drawerWillResizeContents:toSize:` (page 3676)
Invoked when the user resizes the drawer or parent.

Instance Methods

drawerDidClose:

Notifies the delegate that the drawer has closed.

```
- (void)drawerDidClose:(NSNotification *)notification
```

Parameters

notification
An `NSDrawerDidCloseNotification` (page ?) notification sent by the default notification center immediately after the drawer has closed.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSDrawer.h`

drawerDidOpen:

Notifies the delegate that the drawer has opened.

```
- (void)drawerDidOpen:(NSNotification *)notification
```

Parameters

notification
An `NSDrawerDidOpenNotification` (page ?) notification, sent by the default notification center immediately after the drawer has opened.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSDrawer.h`

drawerShouldClose:

Asks the delegate if the specified drawer should close.

```
- (BOOL)drawerShouldClose:(NSDrawer *)sender
```

Parameters

sender

The drawer being closed.

Return Value

`YES` to allow the drawer to close; `NO` to prevent it from closing.

Discussion

This method is invoked on user-initiated attempts to close a drawer by dragging it or when the `close:` (page ?) method is called.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSDrawer.h`

drawerShouldOpen:

Asks the delegate if the specified drawer should open.

```
- (BOOL)drawerShouldOpen:(NSDrawer *)sender
```

Parameters

sender

The drawer requesting permission to open.

Return Value

`YES` if the drawer should open; `NO` to prevent the drawer from opening.

Discussion

This method is invoked on user-initiated attempts to open a drawer by dragging it or when the `open:` (page ?) method is called.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSDrawer.h`

drawerWillClose:

Notifies the delegate the the drawer will close.

```
- (void)drawerWillClose:(NSNotification *)notification
```

Parameters

notification

An `NSDrawerWillCloseNotification` (page ?) notification sent by the default notification center immediately before the drawer is closed.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSDrawer.h

drawerWillOpen:

Notifies the delegate that the drawer will open.

```
- (void)drawerWillOpen:(NSNotification *)notification
```

Parameters

notification
An NSDrawerWillOpenNotification (page ?) notification, sent by the default notification center immediately before the drawer is opened.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSDrawer.h

drawerWillResizeContents:toSize:

Invoked when the user resizes the drawer or parent.

```
- (NSSize)drawerWillResizeContents:(NSDrawer *)sender toSize:(NSSize)contentSize
```

Parameters

sender
The drawer being resized.

contentSize
The proposed new size of the drawer.

Return Value
The size that the drawer should be resized to. To resize to a different size, simply return the desired size from this method; to avoid resizing, return the current size.

Discussion
The receiver's minimum and maximum size constraints have already been applied when this method is invoked. While the user is resizing an NSDrawer or its parent, the delegate is sent a series of windowWillResize messages as the NSDrawer or parent window is dragged.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSDrawer.h

NSEditor Protocol Reference

(informal protocol)

Adopted by	NSController (informal protocol)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSKeyValueBinding.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Cocoa Bindings Programming Topics

Overview

The NSEditor informal protocol is implemented by controllers and user interface elements. It provides a means for requesting that the receiver commit or discard any pending edits.

These methods are typically invoked on user interface elements by a controller. They can also be sent to a controller in response to a user's attempt to save a document or quit an application.

NSController provides an implementation of this protocol, as do the Application Kit user interface elements that support binding.

Tasks

Managing Editing

- `discardEditing` (page 3679)
  Causes the receiver to discard any changes, restoring the previous values.
- `commitEditing` (page 3678)
  Returns whether the receiver was able to commit any pending edits.
- `commitEditingWithDelegate:didCommitSelector:contextInfo:` (page 3678)
  Attempt to commit any currently edited results of the receiver.

Instance Methods

commitEditing

Returns whether the receiver was able to commit any pending edits.

```
- (BOOL)commitEditing
```

Discussion
Returns `YES` if the changes were successfully applied to the model, `NO` otherwise. A commit is denied if the receiver fails to apply the changes to the model object, perhaps due to a validation error.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- discardEditing` (page 3679)

Related Sample Code
AbstractTree
Core Data HTML Store
CoreRecipes
Image Kit with Core Data

Declared In
`NSKeyValueBinding.h`

commitEditingWithDelegate:didCommitSelector:contextInfo:

Attempt to commit any currently edited results of the receiver.

```
- (void)commitEditingWithDelegate:(id)delegate
    didCommitSelector:(SEL)didCommitSelector contextInfo:(void *)contextInfo
```

Discussion
The receiver must have been registered as the editor of an object using `objectDidBeginEditing:`, and has not yet been unregistered by a subsequent invocation of `objectDidEndEditing:`. When the committing has either succeeded or failed, send the following message to the specified object. The *`didCommitSelector`* method must have the following method signature:

```
- (void)editor:(id)editor didCommit:(BOOL)didCommit contextInfo:(void
*)contextInfo
```

If an error occurs while attempting to commit, for example if key-value coding validation fails, an implementation of this method should typically send the NSView in which editing is being done a `presentError:modalForWindow:delegate:didRecoverSelector:contextInfo:` message, specifying the view's containing window.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
CoreRecipes

Declared In
`NSKeyValueBinding.h`

discardEditing

Causes the receiver to discard any changes, restoring the previous values.

```
- (void)discardEditing
```

Availability
Available in Mac OS X v10.3 and later.

See Also
`- commitEditing` (page 3678)

Declared In
`NSKeyValueBinding.h`

NSEditorRegistration Protocol Reference

(informal protocol)

Adopted by	NSController (informal protocol)
Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSKeyValueBinding.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Cocoa Bindings Programming Topics

Overview

The `NSEditorRegistration` informal protocol is implemented by controllers to provide an interface for a view, the editor, to inform the controller when it has uncommitted changes.

An implementor is responsible for tracking which editors have uncommitted changes, and sending those editors `commitEditing` (page 3678) and `discardEditing` (page 3679) messages, as appropriate, to force the editor to submit, or discard, their values.

`NSController` provides an implementation of this informal protocol. You would implement this protocol if you wanted to provide your own controller class without subclassing `NSController`.

Tasks

Managing Editing

- `objectDidBeginEditing:` (page 3682)
  This message should be sent to the receiver when *`editor`* has uncommitted changes that can affect the receiver.
- `objectDidEndEditing:` (page 3682)
  This message should be sent to the receiver when *`editor`* has finished editing a property belonging to the receiver.

Instance Methods

objectDidBeginEditing:

This message should be sent to the receiver when *editor* has uncommitted changes that can affect the receiver.

```
- (void)objectDidBeginEditing:(id)editor
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- objectDidEndEditing: (page 3682)

Declared In
NSKeyValueBinding.h

objectDidEndEditing:

This message should be sent to the receiver when *editor* has finished editing a property belonging to the receiver.

```
- (void)objectDidEndEditing:(id)editor
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- objectDidBeginEditing: (page 3682)

Declared In
NSKeyValueBinding.h

NSFontPanelValidation Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSFontPanel.h
Companion guide	Font Panel

Overview

Informal protocol. The Font Panel can be explicitly ordered to display some or all of its elements by responding to the `validModesForFontPanel:` (page 3683) instance method.

Tasks

Validating Modes for a Font Panel

`- validModesForFontPanel:` (page 3683)
Returns the mode mask corresponding to the expected font panel mode.

Instance Methods

validModesForFontPanel:

Returns the mode mask corresponding to the expected font panel mode.

```
- (NSUInteger)validModesForFontPanel:(NSFontPanel *)fontPanel
```

Discussion
The mode masks are defined in `Mode Masks` (page 3684).

The Font Panel has the ability to hide elements that are not applicable for a given context by having the target respond to `validModesForFontPanel:`. If the target desires a font panel mode other than the standard mode mask, it must respond to this method.

This message is sent up the responder chain to the first responder implementing the method. Ideally that object should be the first responder found that also implements `changeFont:` (page 1239).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSFontPanel.h`

Constants

Mode Masks

The following constants correspond to the available font panel mode masks returned by `validModesForFontPanel:` (page 3683).

```
enum {
    NSFontPanelFaceModeMask = 1 << 0,
    NSFontPanelSizeModeMask = 1 << 1,
    NSFontPanelCollectionModeMask = 1 << 2,
    NSFontPanelUnderlineEffectModeMask = 1<<8,
    NSFontPanelStrikethroughEffectModeMask = 1<<9,
    NSFontPanelTextColorEffectModeMask = 1<< 10,
    NSFontPanelDocumentColorEffectModeMask = 1<<11,
    NSFontPanelShadowEffectModeMask = 1<<12,
    NSFontPanelAllEffectsModeMask = 0XFFF00,
    NSFontPanelStandardModesMask = 0xFFFF,
    NSFontPanelAllModesMask = 0xFFFFFFFF
};
```

Constants

`NSFontPanelFaceModeMask`
Display the typeface column.
Available in Mac OS X v10.3 and later.
Declared in `NSFontPanel.h`.

`NSFontPanelSizeModeMask`
Display the font size column.
Available in Mac OS X v10.3 and later.
Declared in `NSFontPanel.h`.

`NSFontPanelCollectionModeMask`
Display the font collections column.
Available in Mac OS X v10.3 and later.
Declared in `NSFontPanel.h`.

`NSFontPanelUnderlineEffectModeMask`
Display the underline popup menu.
Available in Mac OS X v10.4 and later.
Declared in `NSFontPanel.h`.

`NSFontPanelStrikethroughEffectModeMask`
Display the strike-through popup menu.

Available in Mac OS X v10.4 and later.

Declared in `NSFontPanel.h`.

`NSFontPanelTextColorEffectModeMask`
Display the text color button.

Available in Mac OS X v10.4 and later.

Declared in `NSFontPanel.h`.

`NSFontPanelDocumentColorEffectModeMask`
Display the document color button.

Available in Mac OS X v10.4 and later.

Declared in `NSFontPanel.h`.

`NSFontPanelShadowEffectModeMask`
Display the shadow effects button.

Available in Mac OS X v10.4 and later.

Declared in `NSFontPanel.h`.

`NSFontPanelAllEffectsModeMask`
Display all the effects user interface items.

Available in Mac OS X v10.4 and later.

Declared in `NSFontPanel.h`.

`NSFontPanelStandardModesMask`
Display the standard default font panel—that is, including the collections, typeface, and size columns.

Available in Mac OS X v10.3 and later.

Declared in `NSFontPanel.h`.

`NSFontPanelAllModesMask`
Display all the available adornments.

Available in Mac OS X v10.3 and later.

Declared in `NSFontPanel.h`.

NSGlyphStorage Protocol Reference

Adopted by	NSLayoutManager
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.3 and later.
Declared in	AppKit/NSGlyphGenerator.h
Companion guides	Text System Overview Text Layout Programming Guide

Overview

The `NSGlyphStorage` protocol defines the methods that a glyph storage object must implement in order to interact properly with `NSGlyphGenerator`.

An example of a Cocoa class conforming to the `NSGlyphStorage` protocol is `NSLayoutManager`.

Tasks

Returning Text Storage

`- attributedString` (page 3688) *required method*
Returns the text storage object from which the `NSGlyphGenerator` object procures characters for glyph generation. (required)

Returning Glyph Display Options

`- layoutOptions` (page 3689) *required method*
Returns the current layout options. (required)

Modifying the Glyph Cache

`- insertGlyphs:length:forStartingGlyphAtIndex:characterIndex:` (page 3688) *required method*
Inserts the given glyphs into the glyph cache and maps them to the specified characters. (required)

Instance Methods

attributedString

Returns the text storage object from which the NSGlyphGenerator object procures characters for glyph generation. (required)

```
- (NSAttributedString *)attributedString
```

Return Value
The receiver's text storage object.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSGlyphGenerator.h

insertGlyphs:length:forStartingGlyphAtIndex:characterIndex:

Inserts the given glyphs into the glyph cache and maps them to the specified characters. (required)

```
- (void)insertGlyphs:(const NSGlyph *)glyphs length:(NSUInteger)length
    forStartingGlyphAtIndex:(NSUInteger)glyphIndex
    characterIndex:(NSUInteger)charIndex
```

Parameters

glyphs
The glyphs to insert.

glyphIndex
Location in the glyph cache to begin inserting glyphs.

length
Number of glyphs to insert.

charIndex
Index of first character to be mapped.

Discussion
This is a bulk insert method for the glyph cache.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSGlyphGenerator.h

layoutOptions

Returns the current layout options. (required)

```
- (NSUInteger)layoutOptions
```

Return Value
The layout options as a bit mask, as defined in "Constants" (page 3689).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSGlyphGenerator.h`

setIntAttribute:value:forGlyphAtIndex:

Sets a custom attribute value for a given glyph. (required)

```
- (void)setIntAttribute:(NSInteger)attributeTag value:(NSInteger)val
    forGlyphAtIndex:(NSUInteger)glyphIndex
```

Parameters

attributeTag
The custom attribute.

val
The new attribute value.

glyphIndex
Index of the glyph whose attribute is set.

Discussion
Custom attributes are glyph attributes such as `NSGlyphInscription` or attributes defined by subclasses. Subclasses that define their own custom attributes must override this method and provide their own storage for the attribute values. Nonnegative tags are reserved; you can define your own attributes with negative tags and set values using this method.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSGlyphGenerator.h`

Constants

Layout Options

These constants describe layout options returned as a bit mask by the `layoutOptions` (page 3689) method.

```
enum {
    NSShowControlGlyphs = (1 << 0),
    NSShowInvisibleGlyphs = (1 << 1),
    NSWantsBidiLevels = (1 << 2)
};
```

Constants

`NSShowControlGlyphs`

Generates displayable glyphs for control characters.

Available in Mac OS X v10.3 and later.

Declared in `NSGlyphGenerator.h`.

`NSShowInvisibleGlyphs`

Generates displayable glyphs for invisible characters.

Available in Mac OS X v10.3 and later.

Declared in `NSGlyphGenerator.h`.

`NSWantsBidiLevels`

Generates directional formatting codes for bidirectional text.

Available in Mac OS X v10.3 and later.

Declared in `NSGlyphGenerator.h`.

Declared In

`NSGlyphGenerator.h`

NSIgnoreMisspelledWords Protocol Reference

Adopted by	NSText
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSSpellProtocol.h
Companion guide	Spell Checking

Overview

Implement this protocol to have the Ignore button in the Spelling panel function properly. The Ignore button allows the user to accept a word that the spelling checker believes is misspelled. In order for this action to update the "ignored words" list for the document being checked, the NSIgnoreMisspelledWords protocol must be implemented.

This protocol is necessary because a list of ignored words is useful only if it pertains to the entire document being checked, but the spelling checker (NSSpellChecker object) does not check the entire document for spelling at once. The spelling checker returns as soon as it finds a misspelled word. Thus, it checks only a subset of the document at any one time. The user usually wants to check the entire document, so usually several spelling checks are run in succession until no misspelled words are found. This protocol allows the list of ignored words to be maintained per document, even though the spelling checks are not run per document.

The NSIgnoreMisspelledWords protocol specifies a method, `ignoreSpelling:` (page 3692), which should be implemented like this:

```
- (void)ignoreSpelling:(id)sender {
    [[NSSpellChecker sharedSpellChecker] ignoreWord:[[sender selectedCell]
stringValue]
                                        inSpellDocumentWithTag: myDocumentTag];
}
```

The second argument to the NSSpellChecker method `ignoreWord:inSpellDocumentWithTag:` (page 2527) is a tag that the NSSpellChecker can use to distinguish the documents being checked. Once the NSSpellChecker has a way to distinguish the various documents, it can append new ignored words to the appropriate list.

To make the ignored words feature useful, the application must store a document's ignored words list with the document. See the NSSpellChecker class description for more information.

Tasks

Ignoring Spellings

- ignoreSpelling: (page 3692) *required method*

Instance Methods

ignoreSpelling:

- (void)**ignoreSpelling:**(id)*sender*

Discussion
Implement this action method to allow an application to ignore misspelled words on a document-by-document basis. This message is sent by the NSSpellChecker instance to the object whose text is being checked.

Implement this method by using the code shown in the protocol description.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSSpellProtocol.h

NSImageDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSImage.h
Companion guide	Cocoa Drawing Guide

Overview

The `NSImageDelegate` protocol defines the optional methods implemented by delegates of `NSImage` objects.

Tasks

Responding to Drawing Failure

`- imageDidNotDraw:inRect:` (page 3696)
Sent to the delegate when the image object is unable, for whatever reason, to lock focus on its image or draw in the specified rectangle.

Managing Incremental Loads

`- image:didLoadPartOfRepresentation:withValidRows:` (page 3694)
During incremental loading, this method is called repeatedly to inform the delegate that more of the image data is available.

`- image:didLoadRepresentation:withStatus:` (page 3694)
For incremental loading, this method is invoked when the specified image has been loaded and decompressed as fully as is possible.

`- image:didLoadRepresentationHeader:` (page 3695)
During incremental loading, this method is called once enough data has been read to determine the size of the image.

`- image:willLoadRepresentation:` (page 3695)
For incremental loading, this method is invoked when you first attempt to draw the image or otherwise access the bitmap data.

Instance Methods

image:didLoadPartOfRepresentation:withValidRows:

During incremental loading, this method is called repeatedly to inform the delegate that more of the image data is available.

```
- (void)image:(NSImage *)image didLoadPartOfRepresentation:(NSImageRep *)rep
    withValidRows:(NSInteger)rows
```

Parameters

image
The image object whose contents are being loaded.

rep
The image representation object that is receiving and processing the image data.

rows
The number of rows of data that have been decompressed.

Discussion
This method is optional; incremental loading will continue if the delegate does not implement it.

Availability
Available in Mac OS X v10.2 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSImage.h`

image:didLoadRepresentation:withStatus:

For incremental loading, this method is invoked when the specified image has been loaded and decompressed as fully as is possible.

```
- (void)image:(NSImage *)image didLoadRepresentation:(NSImageRep *)rep
    withStatus:(NSImageLoadStatus)status
```

Parameters

image
The image object whose contents are being loaded.

rep
The image representation object that loaded the image data.

status
The status of the load operation. For a list of possible values, see `NSImageLoadStatus` (page 1378) in *NSImage Class Reference*.

Discussion
The delegate must implement this method if it wants to support the incremental loading of images. In that case, you must also set up the image object to be loaded lazily, by initializing it using the `initByReferencingFile:` (page 1350) or `initByReferencingURL:` (page 1351) method.

If an error occurs during downloading or decompression, the *`status`* parameter is set to `NSImageLoadStatusInvalidData`, `NSImageLoadStatusUnexpectedEOF`, or `NSImageLoadStatusReadError`. If the download was cancelled, the *`status`* parameter is set to `NSImageLoadStatusCancelled`.

Availability
Available in Mac OS X v10.2 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSImage.h`

image:didLoadRepresentationHeader:

During incremental loading, this method is called once enough data has been read to determine the size of the image.

```
- (void)image:(NSImage *)image didLoadRepresentationHeader:(NSImageRep *)rep
```

Parameters
`image`
The image object whose contents are being loaded.
`rep`
The image representation object that is receiving and processing the image data.

Discussion
By the time this method is called, the `NSBitmapImageRep` object specified in the *`rep`* parameter is valid and has allocated the memory needed to store the bitmap. The bitmap itself is filled with the image's background color. This method is optional; incremental loading will continue if the delegate does not implement it.

Availability
Available in Mac OS X v10.2 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSImage.h`

image:willLoadRepresentation:

For incremental loading, this method is invoked when you first attempt to draw the image or otherwise access the bitmap data.

```
- (void)image:(NSImage *)image willLoadRepresentation:(NSImageRep *)rep
```

Parameters
`image`
The image object whose contents need to be loaded.

rep
The image representation object that was accessed.

Discussion
Downloading of the image begins immediately after this method returns. This method is optional; incremental loading will continue if the delegate does not implement it.

Availability
Available in Mac OS X v10.2 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSImage.h`

imageDidNotDraw:inRect:

Sent to the delegate when the image object is unable, for whatever reason, to lock focus on its image or draw in the specified rectangle.

```
- (NSImage *)imageDidNotDraw:(id)sender inRect:(NSRect)aRect
```

Parameters
sender
The `NSImage` object that encountered the problem.

aRect
The rectangle that the image object was attempting to draw.

Return Value
An `NSImage` to draw in place of the one in *sender*, or `nil` if the delegate wants to draw the image itself.

Discussion
The delegate can do one of the following:

- Return another `NSImage` object to draw in the sender's place.
- Draw the image itself and return `nil`,.
- Simply return `nil` to indicate that *sender* should give up on the attempt at drawing the image.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSImage.h`

NSKeyValueBindingCreation Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSKeyValueBinding.h
Companion guides	Cocoa Bindings Programming Topics Cocoa Bindings Reference

Overview

The `NSKeyValueBindingCreation` informal protocol provides methods to create and remove bindings between view objects and controllers or controllers and model objects. In addition, it provides a means for a view subclass to advertise the bindings that it exposes. This informal protocol is implemented by `NSObject` and its methods can be overridden by view and controller subclasses.

When a new binding is created it relates the receiver's binding (for example, a property of the view object) to a property of the observable object specified by a key path. When the value of the specified property of the observable object changes, the receiver is notified using the key-value observing mechanism. A binding also specifies binding options that can further customize how the observing and the observed objects interact.

Bindings are considered to be a property of the object which is bound, and all information related to bindings should be retained by the object. All standard bindings on AppKit objects (views, cells, table columns, controllers) unbind their bindings automatically when they are released, but if you create key-value bindings for other kind of objects, you need to make sure that you remove those bindings when you release them (observed objects don't retain their observers, so controllers/model objects might continue referencing and messaging the objects that were bound to them).

Bindings between objects are typically established in Interface Builder using the Bindings inspector. However, there are times it must be done programmatically, such as when establishing a binding between objects in different nib files.

`NSView` subclasses can expose additional key-value-coding/key-value-observing compliant properties as bindings by calling the class method `exposeBinding:` (page 3698) for each of the properties. This is typically done in the class's `initialize` method. By exposing the bindings that an object supports and creating an Interface Builder palette, you can make instances of your own classes bindable in Interface Builder.

Tasks

Exposing Bindings

`+ exposeBinding:` (page 3698)
Exposes the specified *binding*, advertising its availability.

`- exposedBindings` (page 3699)
Returns an array containing the bindings exposed by the receiver.

Managing Bindings

`- valueClassForBinding:` (page 3701)
Returns the class of the value that will be returned for the specified binding.

`- bind:toObject:withKeyPath:options:` (page 3699)
Establishes a binding between a given property of the receiver and the property of a given object specified by a given key path.

`- optionDescriptionsForBinding:` (page 3700)
Returns an array describing the options for the specified binding.

`- infoForBinding:` (page 3700)
Returns a dictionary describing the receiver's *binding*.

`- unbind:` (page 3701)
Removes a given binding between the receiver and a controller.

Class Methods

exposeBinding:

Exposes the specified *binding*, advertising its availability.

```
+ (void)exposeBinding:(NSString *)binding
```

Parameters

binding
The key path for the property to be exposed.

Discussion
The bound property will be accessed using key-value-coding compliant methods. This method is typically invoked in the class's `initialize` implementation.

Bindings exposed using `exposeBinding` will be exposed automatically in exposedBindings (page 3699) unless that method explicitly filters them out, for example in subclasses.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSKeyValueBinding.h

Instance Methods

bind:toObject:withKeyPath:options:

Establishes a binding between a given property of the receiver and the property of a given object specified by a given key path.

```
- (void)bind:(NSString *)binding
    toObject:(id)observableController
    withKeyPath:(NSString *)keyPath
    options:(NSDictionary *)options
```

Parameters

binding

The key path for a property of the receiver previously exposed using the `exposeBinding:` (page 3698) method.

observableController

The bound-to object.

keyPath

A key path to a property reachable from *observableController*. The elements in the path must be key-value observing compliant (see *Key-Value Observing Programming Guide*).

options

A dictionary containing options for the binding, such as placeholder objects or an `NSValueTransformer` identifier as described in "Constants" (page 3702). This value is optional—pass `nil` to specify no options.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- unbind:` (page 3701)

Related Sample Code
AnimatedTableView
Simple Bindings Adoption
Sketch+Accessibility

Declared In
NSKeyValueBinding.h

exposedBindings

Returns an array containing the bindings exposed by the receiver.

```
- (NSArray *)exposedBindings
```

Return Value
An array containing the bindings exposed by the receiver.

Discussion
A subclass can override this method to remove bindings that are exposed by a superclass that are not appropriate for the subclass.

Availability
Available in Mac OS X v10.3 and later.

See Also
`+ exposeBinding:` (page 3698)

Declared In
`NSKeyValueBinding.h`

infoForBinding:

Returns a dictionary describing the receiver's *binding*.

```
- (NSDictionary *)infoForBinding:(NSString *)binding
```

Parameters

binding

The name of a binding.

Return Value
A dictionary with information about *binding*, or `nil` if the binding is not bound. The dictionary contains three key/value pairs: `NSObservedObjectKey`: object bound, `NSObservedKeyPathKey`: key path bound, `NSOptionsKey`: dictionary with the options and their values for the bindings.

Discussion
This method is mostly for use by subclasses which want to analyze the existing bindings of an object.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSKeyValueBinding.h`

optionDescriptionsForBinding:

Returns an array describing the options for the specified binding.

```
- (NSArray *)optionDescriptionsForBinding:(NSString *)binding
```

Parameters

binding

The name of a binding

Return Value
Returns an array of `NSAttributeDescription` that describe the options for *binding*.

Discussion
The `NSAttributeDescription` instances in the array are used by Interface Builder to build the options editor user interface of the bindings inspector.

- The option name displayed for the option in the bindings inspector is based on the value of the `NSAttributeDescription` method `name`.
- The type of editor displayed for the option in the bindings inspector is based on the value of the `NSAttributeDescription` method `attributeType`.
- The default value displayed in the bindings inspector for the option is based on the value of the `NSAttributeDescription` method `defaultValue`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSKeyValueBinding.h`

unbind:

Removes a given binding between the receiver and a controller.

```
- (void)unbind:(NSString *)binding
```

Parameters
binding
The name of a binding.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- bind:toObject:withKeyPath:options:` (page 3699)

Declared In
`NSKeyValueBinding.h`

valueClassForBinding:

Returns the class of the value that will be returned for the specified binding.

```
- (Class)valueClassForBinding:(NSString *)binding
```

Parameters
binding
The name of a binding.

Return Value
The class of the value that will be returned for *binding*.

Discussion
This method is used by Interface Builder to determine the appropriate transformers for a binding.

Implementation of this method is optional.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSKeyValueBinding.h`

Constants

Binding Options

The following values are used as keys in the options dictionary passed to the `bind:toObject:withKeyPath:options:` (page 3699) method. These keys are also used in the dictionary returned as the `NSOptionsKey` value of `infoForBinding:` (page 3700). See the *Cocoa Bindings Reference* for more information.

```
NSString *NSAllowsEditingMultipleValuesSelectionBindingOption;
NSString *NSAllowsNullArgumentBindingOption;
NSString *NSAlwaysPresentsApplicationModalAlertsBindingOption;
NSString *NSConditionallySetsEditableBindingOption;
NSString *NSConditionallySetsEnabledBindingOption;
NSString *NSConditionallySetsHiddenBindingOption;
NSString *NSContinuouslyUpdatesValueBindingOption;
NSString *NSCreatesSortDescriptorBindingOption;
NSString *NSDeletesObjectsOnRemoveBindingsOption;
NSString *NSDisplayNameBindingOption;
NSString *NSDisplayPatternBindingOption;
NSString *NSContentPlacementTagBindingOption;
NSString *NSHandlesContentAsCompoundValueBindingOption;
NSString *NSInsertsNullPlaceholderBindingOption;
NSString *NSInvokesSeparatelyWithArrayObjectsBindingOption;
NSString *NSMultipleValuesPlaceholderBindingOption;
NSString *NSNoSelectionPlaceholderBindingOption;
NSString *NSNotApplicablePlaceholderBindingOption;
NSString *NSNullPlaceholderBindingOption;
NSString *NSRaisesForNotApplicableKeysBindingOption;
NSString *NSPredicateFormatBindingOption;
NSString *NSSelectorNameBindingOption;
NSString *NSSelectsAllWhenSettingContentBindingOption;
NSString *NSValidatesImmediatelyBindingOption;
NSString *NSValueTransformerNameBindingOption;
NSString *NSValueTransformerBindingOption;
```

Constants

`NSAllowsEditingMultipleValuesSelectionBindingOption`

An `NSNumber` object containing a Boolean value that determines if the binding allows editing when the value represents a multiple selection.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSAlwaysPresentsApplicationModalAlertsBindingOption`
An `NSNumber` object containing a Boolean value that determines if validation and error alert panels displayed as a result of this binding are displayed as application modal alerts. If `YES`, then the alerts are displayed application model, otherwise they are displayed as sheets.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSAllowsNullArgumentBindingOption`
An `NSNumber` object containing a Boolean value that determines if the argument bindings allows passing argument values of `nil`.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSConditionallySetsEditableBindingOption`
An `NSNumber` object containing a Boolean value that determines if the editable state of the user interface item is automatically configured based on the controller's selection.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSConditionallySetsEnabledBindingOption`
An `NSNumber` object containing a Boolean value that determines if the enabled state of the user interface item is automatically configured based on the controller's selection.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSConditionallySetsHiddenBindingOption`
An `NSNumber` object containing a Boolean value that determines if the hidden state of the user interface item is automatically configured based on the controller's selection.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSContentPlacementTagBindingOption`
An `NSNumber` object specifying the tag id of the popup menu item to replace with the content of the array. This allows you to use a popup menu that contains both static and bindings generated items.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSContinuouslyUpdatesValueBindingOption`
An `NSNumber` object containing a Boolean value that determines whether the value of the binding is updated as edits are made to the user interface item or is updated only when the user interface item resigns as the responder.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSCreatesSortDescriptorBindingOption`
An `NSNumber` object containing a Boolean value that determines if a sort descriptor is created for a table column.

If this value is `NO`, then the table column does not allow sorting.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSDeletesObjectsOnRemoveBindingsOption`
An `NSNumber` object containing a Boolean value that determines if an object is deleted from the managed context immediately upon being removed from a relationship.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSDisplayNameBindingOption`
An `NSString` object containing a human readable string to be displayed for a predicate.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSDisplayPatternBindingOption`
An `NSString` object that specifies a format string used to construct the final value of a string.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSHandlesContentAsCompoundValueBindingOption`
An `NSNumber` object containing a Boolean value that determines if the content is treated as a compound value.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSInsertsNullPlaceholderBindingOption`
An `NSNumber` object containing a Boolean value that determines if an additional item which represents `nil` is inserted into a matrix or pop-up menu before the items in the content array.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSInvokesSeparatelyWithArrayObjectsBindingOption`
An `NSNumber` object containing a Boolean value that determines whether the specified selector is invoked with the array as the argument or is invoked repeatedly with each array item as an argument.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSMultipleValuesPlaceholderBindingOption`
An object that is used as a placeholder when the key path of the bound controller returns the `NSMultipleValuesMarker` marker for a binding.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSNoSelectionPlaceholderBindingOption`
An object that is used as a placeholder when the key path of the bound controller returns the `NSNoSelectionMarker` marker for a binding.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSNotApplicablePlaceholderBindingOption`
An object that is used as a placeholder when the key path of the bound controller returns the `NSNotApplicableMarker` marker for a binding.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSNullPlaceholderBindingOption`
An object that is used as a placeholder when the key path of the bound controller returns `nil` for a binding.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSPredicateFormatBindingOption`
An `NSString` object containing the predicate pattern string for the predicate bindings. Use $value to refer to the value in the search field.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSRaisesForNotApplicableKeysBindingOption`
An `NSNumber` object containing a Boolean value that specifies if an exception is raised when the binding is bound to a key that is not applicable—for example when an object is not key-value coding compliant for a key.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectorNameBindingOption`
An `NSString` object that specifies the method selector invoked by the target binding when the user interface item is clicked.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectsAllWhenSettingContentBindingOption`
An `NSNumber` object containing a Boolean value that specifies if all the items in the array controller are selected when the content is set.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSValidatesImmediatelyBindingOption`
An `NSNumber` object containing a Boolean value that determines if the contents of the binding are validated immediately.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSValueTransformerNameBindingOption`
The value for this key is an identifier of a registered `NSValueTransformer` instance that is applied to the bound value.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSValueTransformerBindingOption`
An `NSValueTransformer` instance that is applied to the bound value.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

Binding Dictionary Keys

The following values are used as keys in the dictionary returned by `infoForBinding:` (page 3700)

```
NSString *NSObservedObjectKey;
NSString *NSObservedKeyPathKey;
NSString *NSOptionsKey;
```

Constants

`NSObservedObjectKey`

The object that is the observable controller of the binding.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSObservedKeyPathKey`

An `NSString` object containing the key path of the binding.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSOptionsKey`

An `NSDictionary` object containing key value pairs as specified in the options dictionary when the binding was created.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

Bindings

The following values are used to specify a binding to `bind:toObject:withKeyPath:options:` (page 3699), `infoForBinding:` (page 3700), `unbind:` (page 3701) and `valueClassForBinding:` (page 3701). See *Cocoa Bindings Reference* for more information.

```
NSString *NSAlignmentBinding;
NSString *NSAlternateImageBinding;
NSString *NSAlternateTitleBinding;
NSString *NSAnimateBinding;
NSString *NSAnimationDelayBinding;
NSString *NSArgumentBinding;
NSString *NSAttributedStringBinding;
NSString *NSContentArrayBinding;
NSString *NSContentArrayForMultipleSelectionBinding;
NSString *NSContentBinding;
NSString *NSContentDictionaryBinding;
NSString *NSContentHeightBinding;
NSString *NSContentObjectBinding;
NSString *NSContentObjectsBinding;
NSString *NSContentSetBinding;
NSString *NSContentValuesBinding;
NSString *NSContentWidthBinding;
NSString *NSCriticalValueBinding;
NSString *NSDataBinding;
NSString *NSDisplayPatternTitleBinding;
NSString *NSDisplayPatternValueBinding;
NSString *NSDocumentEditedBinding;
NSString *NSDoubleClickArgumentBinding;
NSString *NSDoubleClickTargetBinding;
NSString *NSEditableBinding;
NSString *NSEnabledBinding;
NSString *NSExcludedKeysBinding;
NSString *NSFilterPredicateBinding;
NSString *NSFontBinding;
NSString *NSFontBoldBinding;
NSString *NSFontFamilyNameBinding;
NSString *NSFontItalicBinding;
NSString *NSFontNameBinding;
NSString *NSFontSizeBinding;
NSString *NSHeaderTitleBinding;
NSString *NSHiddenBinding;
NSString *NSImageBinding;
NSString *NSIncludedKeysBinding;
NSString *NSInitialKeyBinding;
NSString *NSInitialValueBinding;
NSString *NSIsIndeterminateBinding;
NSString *NSLabelBinding;
NSString *NSLocalizedKeyDictionaryBinding;
NSString *NSManagedObjectContextBinding;
NSString *NSMaximumRecentsBinding;
NSString *NSMaxValueBinding;
NSString *NSMaxWidthBinding;
NSString *NSMinValueBinding;
NSString *NSMinWidthBinding;
NSString *NSMixedStateImageBinding;
NSString *NSOffStateImageBinding;
NSString *NSOnStateImageBinding;
NSString *NSPredicateBinding;
NSString *NSRecentSearchesBinding;
NSString *NSRepresentedFilenameBinding;
NSString *NSRowHeightBinding;
NSString *NSSelectedIdentifierBinding;
NSString *NSSelectedIndexBinding;
```

```
NSString *NSSelectedLabelBinding;
NSString *NSSelectedObjectBinding;
NSString *NSSelectedObjectsBinding;
NSString *NSSelectedTagBinding;
NSString *NSSelectedValueBinding;
NSString *NSSelectedValuesBinding;
NSString *NSSelectionIndexesBinding;
NSString *NSSelectionIndexPathsBinding;
NSString *NSSortDescriptorsBinding;
NSString *NSTargetBinding;
NSString *NSTextColorBinding;
NSString *NSTitleBinding;
NSString *NSToolTipBinding;
NSString *NSTransparentBinding;
NSString *NSValueBinding;
NSString *NSValuePathBinding;
NSString *NSValueURLBinding;
NSString *NSVisibleBinding;
NSString *NSWarningValueBinding;
NSString *NSWidthBinding;
```

Constants

`NSAlignmentBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSAlternateImageBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSAlternateTitleBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSAnimateBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSAnimationDelayBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSArgumentBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSAttributedStringBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSContentArrayBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSContentArrayForMultipleSelectionBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSContentBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSContentDictionaryBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSContentHeightBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSContentObjectBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSContentObjectsBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSContentSetBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSContentValuesBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSContentWidthBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSCriticalValueBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSDataBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSDisplayPatternTitleBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSDisplayPatternValueBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSDocumentEditedBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSDoubleClickArgumentBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSDoubleClickTargetBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSEditableBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSEnabledBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSExcludedKeysBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSFilterPredicateBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSFontBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSFontBoldBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSFontFamilyNameBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSFontItalicBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSFontNameBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSFontSizeBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSHeaderTitleBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSHiddenBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSImageBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSIncludedKeysBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSInitialKeyBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSInitialValueBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSIsIndeterminateBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSLabelBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSLocalizedKeyDictionaryBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSManagedObjectContextBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSMaximumRecentsBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSMaxValueBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSMaxWidthBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSMinValueBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSMinWidthBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSMixedStateImageBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSOffStateImageBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSOnStateImageBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSPredicateBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSRecentSearchesBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSRepresentedFilenameBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSRowHeightBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectedIdentifierBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectedIndexBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectedLabelBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectedObjectBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectedObjectsBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectedTagBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectedValueBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectedValuesBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectionIndexesBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSelectionIndexPathsBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSSortDescriptorsBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSTargetBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSTextColorBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSTitleBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSToolTipBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSTransparentBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.5 and later.

Declared in `NSKeyValueBinding.h`.

`NSValueBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSValuePathBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSValueURLBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSVisibleBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSWarningValueBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

`NSWidthBinding`

See *Cocoa Bindings Reference* for more information.

Available in Mac OS X v10.4 and later.

Declared in `NSKeyValueBinding.h`.

NSLayoutManagerDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSLayoutManager.h
Companion guides	Text System Overview Text Layout Programming Guide

Overview

The `NSLayoutManagerDelegate` protocol defines the optional methods implemented by delegates of `NSLayoutManager` objects.

Tasks

Invalidating Glyphs and Layout

`- layoutManagerDidInvalidateLayout:` (page 3719)
Informs the delegate that the given layout manager has invalidated layout information (not glyph information).

Handling Layout for Text Containers

`- layoutManager:didCompleteLayoutForTextContainer:atEnd:` (page 3718)
Informs the delegate that the given layout manager has finished laying out text in the given text container.

Managing Temporary Attribute Support

`- layoutManager:shouldUseTemporaryAttributes:forDrawingToScreen:atCharacterIndex:effectiveRange:` (page 3718)

Sent when the layout manager is drawing and needs to decide whether or not to use temporary attributes.

Instance Methods

layoutManager:didCompleteLayoutForTextContainer:atEnd:

Informs the delegate that the given layout manager has finished laying out text in the given text container.

```
- (void)layoutManager:(NSLayoutManager *)aLayoutManager
    didCompleteLayoutForTextContainer:(NSTextContainer *)aTextContainer
    atEnd:(BOOL)flag
```

Parameters

aLayoutManager

The layout manager doing the layout.

aTextContainer

The text container in which layout is complete. If `nil`, if there aren't enough containers to hold all the text; the delegate can use this information as a cue to add another text container.

flag

If `YES`, *aLayoutManager* is finished laying out its text—this also means that *aTextContainer* is the final text container used by the layout manager. Delegates can use this information to show an indicator or background or to enable or disable a button that forces immediate layout of text.

Discussion

This message is sent whenever a text container has been filled. This method can be useful for paginating.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSLayoutManager.h`

layoutManager:shouldUseTemporaryAttributes:forDrawingToScreen:atCharacterIndex:effectiveRange:

Sent when the layout manager is drawing and needs to decide whether or not to use temporary attributes.

```
- (NSDictionary *)layoutManager:(NSLayoutManager *)layoutManager
    shouldUseTemporaryAttributes:(NSDictionary *)attrs
    forDrawingToScreen:(BOOL)toScreen atCharacterIndex:(NSUInteger)charIndex
    effectiveRange:(NSRangePointer)effectiveCharRange
```

Parameters

layoutManager

The layout manager sending the message.

attrs

The temporary attributes currently in effect for the given character range.

toScreen

YES if the layout manager is drawing to the screen; otherwise, NO.

charIndex

Index of the first character in the range being drawn.

effectiveCharRange

On input and output, the effective range to which the temporary attributes apply.

Return Value

The temporary attributes for the layout manager to use, or nil if no temporary attributes are to be used.

Discussion

The default behavior, if this method is not implemented, is to use temporary attributes only when drawing to the screen, so an implementation to match that behavior would return *attrs* if *toScreen* is YES and nil otherwise, without changing *effectiveCharRange*.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSLayoutManager.h

layoutManagerDidInvalidateLayout:

Informs the delegate that the given layout manager has invalidated layout information (not glyph information).

```
- (void)layoutManagerDidInvalidateLayout:(NSLayoutManager *)sender
```

Parameters

sender

The layout manager that invalidated layout.

Discussion

This method is invoked only when layout was complete and then became invalidated for some reason. Delegates can use this information to show an indicator of background layout or to enable a button that forces immediate layout of text.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSLayoutManager.h

NSMatrixDelegate Protocol Reference

Conforms to	NSControlTextEditingDelegate
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSMatrix.h
Companion guide	Matrix Programming Guide

Overview

The `NSMatrixDelegate` protocol defines the optional methods implemented by delegates of `NSMatrix` objects.

This protocol simply adopts the `NSControlTextEditingDelegate` protocol, adding no additional methods. See *NSControlTextEditingDelegate Protocol Reference* for more information.

NSMenuDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSMenu.h
Companion guide	Application Menu and Pop-up List Programming Topics

Overview

The `NSMenuDelegate` protocol defines the optional methods implemented by delegates of `NSMenu` objects.

Tasks

Handling Keyboard Equivalents

`- menuHasKeyEquivalent:forEvent:target:action:` (page 3726)
Invoked to allow the delegate to return the target and action for a key-down event.

Updating Menu Layout

`- menu:updateItem:atIndex:shouldCancel:` (page 3725)
Invoked to let the delegate update a menu item before it is displayed.

`- confinementRectForMenu:onScreen:` (page 3724)
Invoked to allow the delegate to specify a display location for the menu.

Handling Highlighting

`- menu:willHighlightItem:` (page 3725)
Invoked to indicates that a menu is about to highlight a given item.

Handling Open and Close Events

- `menuWillOpen:` (page 3727)
  Invoked when a menu is about to open.
- `menuDidClose:` (page 3726)
  Invoked after a menu closed.

Handling Tracking

- `numberOfItemsInMenu:` (page 3728)
  Invoked when a menu is about to be displayed at the start of a tracking session so the delegate can specify the number of items in the menu.
- `menuNeedsUpdate:` (page 3727)
  Invoked when a menu is about to be displayed at the start of a tracking session so the delegate can modify the menu.

Instance Methods

confinementRectForMenu:onScreen:

Invoked to allow the delegate to specify a display location for the menu.

```
- (NSRect)confinementRectForMenu:(NSMenu *)menu onScreen:(NSScreen *)screen
```

Parameters

menu
The menu object.

screen
The screen the menu will open on.

Return Value
The rectangle, in screen coordinates, the menu should be displayed within.

Discussion
This method is sent to the delegate when a menu is about to be opened on the specified screen.

If you return `NSZeroRect`, or if the delegate does not implement this method, the menu will be confined to the bounds appropriate for the given screen. The returned rect may not be honored in all cases, for example, if it would force the menu to be too small.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSMenu.h`

menu:updateItem:atIndex:shouldCancel:

Invoked to let the delegate update a menu item before it is displayed.

```
- (BOOL)menu:(NSMenu *)menu updateItem:(NSMenuItem *)item atIndex:(NSInteger)index
    shouldCancel:(BOOL)shouldCancel
```

Parameters

menu

The menu object that owns `item`.

item

The menu-item object that may be updated.

index

The integer index of the menu item.

shouldCancel

Set to `YES` if, due to some user action, the menu no longer needs to be displayed before all the menu items have been updated. You can ignore this flag, return `YES`, and continue; or you can save your work (to save time the next time your delegate is called) and return `NO` to stop the updating.

Return Value

`YES` to continue the process. If you return `NO`, your `menu:updateItem:atIndex:shouldCancel:` is not called again. In that case, it is your responsibility to trim any extra items from the menu.

Discussion

If your `numberOfItemsInMenu:` (page 3728) delegate method returns a positive value, then your `menu:updateItem:atIndex:shouldCancel:` method is called for each item in the menu. You can then update the menu title, image, and so forth for each menu item.

Availability

Available in Mac OS X v10.3 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSMenu.h`

menu:willHighlightItem:

Invoked to indicates that a menu is about to highlight a given item.

```
- (void)menu:(NSMenu *)menu willHighlightItem:(NSMenuItem *)item
```

Parameters

menu

The menu object about to highlight an item.

item

The item about to be highlighted.

Discussion

Only one item per menu can be highlighted at a time. If *item* is `nil`, it means that all items in the menu are about to be unhighlighted

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `highlightedItem` (page 1614) (NSMenu)

Declared In
`NSMenu.h`

menuDidClose:

Invoked after a menu closed.

```
- (void)menuDidClose:(NSMenu *)menu
```

Parameters

menu
The menu that closed.

Special Considerations

Do not modify the structure of the menu or the menu items during this method.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `menuWillOpen:` (page 3727)

Declared In
`NSMenu.h`

menuHasKeyEquivalent:forEvent:target:action:

Invoked to allow the delegate to return the target and action for a key-down event.

```
- (BOOL)menuHasKeyEquivalent:(NSMenu *)menu forEvent:(NSEvent *)event target:(id
    *)target action:(SEL *)action
```

Parameters

menu
The menu object sending the delegation message.

event
An `NSEvent` object representing a key-down event.

target
Return by reference the target object for the menu item that corresponds to the event. Specify `nil` to requests the menu's target.

action
Return by reference the action selector for the menu item that corresponds to the event.

Return Value
If there is a valid and enabled menu item that corresponds to this key-down even, return `YES` after specifying the target and action. Return `NO` if there are no items with that key equivalent or if the item is disabled.

Discussion
If the delegate does not define this method, the menu is populated to find out if any items have a matching key equivalent.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `performKeyEquivalent:` (page 1625) (NSMenu)
- `performActionForItemAtIndex:` (page 1624) (NSMenu)

Declared In
`NSMenu.h`

menuNeedsUpdate:

Invoked when a menu is about to be displayed at the start of a tracking session so the delegate can modify the menu.

```
- (void)menuNeedsUpdate:(NSMenu *)menu
```

Parameters

menu
The menu object that is about to be displayed.

Discussion
You can change the menu by adding, removing or modifying menu items. Be sure to set the proper enable state for any new menu items. If populating the menu will take a long time, implement `numberOfItemsInMenu:` (page 3728) and `menu:updateItem:atIndex:shouldCancel:` (page 3725) instead.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSMenu.h`

menuWillOpen:

Invoked when a menu is about to open.

```
- (void)menuWillOpen:(NSMenu *)menu
```

Parameters

menu
The menu that is about to open.

Special Considerations

Do not modify the structure of the menu or the menu items during this method.

Availability
Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `menuDidClose:` (page 3726)

Declared In
`NSMenu.h`

numberOfItemsInMenu:

Invoked when a menu is about to be displayed at the start of a tracking session so the delegate can specify the number of items in the menu.

```
- (NSInteger)numberOfItemsInMenu:(NSMenu *)menu
```

Parameters

menu

The menu object about to be displayed.

Return Value
The number of menu items in the menu.

Discussion
If you return a positive value, the menu is resized by either removing or adding items. Newly created items are blank. After the menu is resized, your `menu:updateItem:atIndex:shouldCancel:` method is called for each item. If you return a negative value, the number of items is left unchanged and `menu:updateItem:atIndex:shouldCancel:` (page 3725) is not called. If you can populate the menu quickly, you can implement `menuNeedsUpdate:` (page 3727) instead of `numberOfItemsInMenu:` and `menu:updateItem:atIndex:shouldCancel:`.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSMenu.h`

NSMenuValidation Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSMenu.h
Companion guide	Application Menu and Pop-up List Programming Topics

Overview

This informal protocol allows your application to update the enabled or disabled status of an `NSMenuItem` object. It declares only one method, `validateMenuItem:` (page 3729).

Tasks

Validating Menu Items

`- validateMenuItem:` (page 3729)
Implemented to override the default action of enabling or disabling a specific menu item.

Instance Methods

validateMenuItem:

Implemented to override the default action of enabling or disabling a specific menu item.

```
- (BOOL)validateMenuItem:(NSMenuItem *)menuItem
```

Parameters

menuItem
An NSMenuItem object that represents the menu item.

Return Value
`YES` to enable *menuItem*, `NO` to disable it.

Discussion
The object implementing this method must be the target of *menuItem*. You can determine which menu item *menuItem* is by querying it for its tag or action.

The following example disables the menu item associated with the `nextRecord` action method when the selected line in a table view is the last one; conversely, it disables the menu item with `priorRecord` as its action method when the selected row is the first one in the table view. (The `countryKeys` array contains names that appear in the table view.)

```
- (BOOL)validateMenuItem:(NSMenuItem *)item {
    int row = [tableView selectedRow];
    if ([item action] == @selector(nextRecord) &&
        (row == [countryKeys indexOfObject:[countryKeys lastObject]])) {
        return NO;
    }
    if ([item action] == @selector(priorRecord) && row == 0) {
        return NO;
    }
    return YES;
}
```

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTAudioContextInsert
QTAudioExtractionPanel
QTKitImport
QTKitPlayer

Declared In
`NSMenu.h`

NSNibAwaking Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSNibLoading.h
Companion guide	Resource Programming Guide

Overview

This informal protocol consists of a single method, `awakeFromNib` (page 3731). Classes can implement this method to initialize state information after objects have been loaded from an Interface Builder archive (nib file).

Tasks

Responding to Being Loaded from a Nib File

- `awakeFromNib` (page 3731)
  Prepares the receiver for service after it has been loaded from an Interface Builder archive, or nib file.

Instance Methods

awakeFromNib

Prepares the receiver for service after it has been loaded from an Interface Builder archive, or nib file.

```
- (void)awakeFromNib
```

Discussion
An `awakeFromNib` message is sent to each object loaded from the archive, but only if it can respond to the message, and only after all the objects in the archive have been loaded and initialized. When an object receives an `awakeFromNib` message, it is guaranteed to have all its outlet instance variables set.

Note: During Interface Builder's test mode, this method is also sent to objects instantiated from loaded palettes, which include executable code for the objects. It is not sent to objects created using the Classes display of the nib file window in Interface Builder.

During the instantiation process, each object in the archive is unarchived and then initialized with the method befitting its type. Cocoa views (and custom views that can be customized using an associated Interface Builder palette) are initialized using their `initWithCoder:` method. Custom views are initialized using their `initWithFrame:` method. Custom classes that have been instantiated in the nib are initialized using their `init` method.

Once all objects have been instantiated and initialized from the archive, the nib loading code attempts to reestablish the connections between each object's outlets and the corresponding target objects. If your custom objects have outlets, an `NSNib` object attempts to reestablish any connections you created in Interface Builder. It starts by trying to establish the connections using your object's own methods first. For each outlet that needs a connection, the `NSNib` object looks for a method of the form `set`*OutletName*`:` in your object. If that method exists, the `NSNib` object calls it, passing the target object as a parameter. If you did not define a setter method with that exact name, the NSNib object searches the object for an instance variable (of type `IBOutlet id`) with the corresponding outlet name and tries to set its value directly. If an instance variable with the correct name cannot be found, initialization of that connection does not occur. Finally, after all the objects are fully initialized, each receives an `awakeFromNib` message.

Important: Because the order in which objects are instantiated from an archive is not guaranteed, your initialization methods should not send messages to other objects in the hierarchy. Messages to other objects can be sent safely from within `awakeFromNib`—by which time it's assured that all the objects are unarchived and initialized (though not necessarily awakened, of course).

Typically, you implement `awakeFromNib` for the class you associate with the "File's Owner" of the nib file. You might also want to implement this method for any other classes you instantiate directly in your nib file. The job of these objects is to give you a hook for connecting the nib file objects to other objects in your application. Once that job is finished, you can either dispose of the objects or use them as a controller for the nib file objects.

An example of how you might use `awakeFromNib` is shown below. Suppose your nib file has two custom views that must be positioned relative to each other at runtime. Trying to position them at initialization time might fail because the other view might not yet be unarchived and initialized yet. However, you can position both of them in the nib file owner's `awakeFromNib` method. In the code below, `firstView` and `secondView` are outlets of the file's owner:

```
- (void)awakeFromNib {
    NSRect viewFrame;

    viewFrame = [firstView frame];
    viewFrame.origin.x += viewFrame.size.width;
    [secondView setFrame:viewFrame];
    return;
}
```

It is recommended that you maintain a one-to-one correspondence between your File's Owner objects and their associated nib files. Loading two nib files with the same File's Owner object causes that object's `awakeFromNib` method being called twice, which could cause some data structures to be reinitialized in undesired ways. It is also recommended that you avoid loading other nib files from your `awakeFromNib` method implementation.

You should call the `super` implementation of `awakeFromNib` only if you know for certain that your superclass provides an implementation. Because the Application Kit does not provide a default implementation of the `awakeFromNib` method, calling `super` results in an exception if the parent class does not implement it. Classes whose immediate parent class is `NSObject` or `NSView` do not need to call the `super` implementation. For any other classes, you can use the `instancesRespondToSelector:` class method of `NSObject` to determine if the parent class responds to `awakeFromNib` and call the method if it does.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ loadNibNamed:owner:` (page 465) (NSBundle Additions)
`- awakeAfterUsingCoder` (NSObject class)
`+ instancesRespondToSelector:` (NSObject class)
`- initWithCoder:` (NSCoding protocol)
`+ initialize` (NSObject class)

Related Sample Code
ImageKitDemo
MenuItemView
MyPhoto
NumberInput_IMKit_Sample
ViewController

Declared In
`NSNibLoading.h`

NSOpenSavePanelDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSSavePanel.h
Companion guides	Application File Management Sheet Programming Topics
Related sample code	QTMetadataEditor

Overview

The `NSOpenSavePanelDelegate` protocol defines the methods that a delegate of `NSOpenPanel` or `NSSavePanel` should implement.

Tasks

Running Panels

- `panel:shouldEnableURL:` (page 3736)
  For `NSOpenPanel` delegates, asks the delegate whether the specified URL should be enabled in the panel. This method is not called for `NSSavePanel` delegates; all URLs are always disabled.
- `panel:validateURL:error:` (page 3737)
  For `NSSavePanel` delegates, asks the delegate for file URL validation when the user chooses the Save button. For `NSOpenPanel` delegates, asks the delegate for file URL validation once for each selected filename (or directory) when the user chooses the Open button.
- `panel:didChangeToDirectoryURL:` (page 3736)
  Informs the delegate that the user changed the selected directory to the directory located at the specified URL. The URL may be `nil` if the current URL can't be represented by an `NSURL` object.
- `panel:userEnteredFilename:confirmed:` (page 3737)
  Tells the delegate that the user confirmed a filename choice by clicking Save in a Save panel.

- `panel:willExpand:` (page 3738)
  Tells the delegate that the Save panel is about to expand or collapse because the user clicked the disclosure triangle that displays or hides the file browser.
- `panelSelectionDidChange:` (page 3738)
  Tells the delegate that the user changed the selection in the specified Save panel.

Instance Methods

panel:didChangeToDirectoryURL:

Informs the delegate that the user changed the selected directory to the directory located at the specified URL. The URL may be `nil` if the current URL can't be represented by an `NSURL` object.

```
- (void)panel:(id)sender
    didChangeToDirectoryURL:(NSURL *)url
```

Parameters

sender
The panel whose directory changed.

url
The URL of the new directory, or `nil` if it can't be represented by an `NSURL` object.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSSavePanel.h`

panel:shouldEnableURL:

For `NSOpenPanel` delegates, asks the delegate whether the specified URL should be enabled in the panel. This method is not called for `NSSavePanel` delegates; all URLs are always disabled.

```
- (BOOL)panel:(id)sender
    shouldEnableURL:(NSURL *)url
```

Parameters

sender
The panel asking whether the URL should be enabled.

url
The URL to be checked.

Return Value
`YES` to allow the URL to be enabled in the panel; otherwise, `NO`.

Discussion
Implementations of this method should be fast to avoid stalling the user interface. Use `panel:validateURL:error:` (page 3737) instead if processing will take a long time.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSSavePanel.h

panel:userEnteredFilename:confirmed:

Tells the delegate that the user confirmed a filename choice by clicking Save in a Save panel.

```
- (NSString *)panel:(id)sender
    userEnteredFilename:(NSString *)filename
    confirmed:(BOOL)okFlag
```

Parameters

sender
The panel reporting the user's confirmation of a filename choice.

filename
The user's filename choice.

okFlag
If YES, the user clicked the Save button; if NO, the user did not.

Return Value
You can either leave the filename alone, return a new filename, or return nil to cancel the save (and leave the Save panel as is). This method is called before any required extension is appended to the filename and before the Save panel asks the user to replace an existing file, if applicable.

Note that in the future, this method may be called multiple times in the sessions as the user types. In those cases, *okFlag* will be NO until the user confirms the choice, in which case *okFlag* will become YES. If the delegate does extensive validation or puts up alerts, it should do so only when *okFlag* is YES.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- panel:validateURL:error: (page 3737)

Declared In
NSSavePanel.h

panel:validateURL:error:

For NSSavePanel delegates, asks the delegate for file URL validation when the user chooses the Save button. For NSOpenPanel delegates, asks the delegate for file URL validation once for each selected filename (or directory) when the user chooses the Open button.

```
- (BOOL)panel:(id)sender
    validateURL:(NSURL *)url
    error:(NSError **)outError
```

Parameters

sender

The panel requesting URL validation.

url

The URL to be validated.

outError

If an error occurred during validation, the error that occurred.

Return Value

`YES` if the URL is an acceptable URL to save to or to open; otherwise, `NO`.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSSavePanel.h`

panel:willExpand:

Tells the delegate that the Save panel is about to expand or collapse because the user clicked the disclosure triangle that displays or hides the file browser.

```
- (void)panel:(id)sender
    willExpand:(BOOL)expanding
```

Parameters

sender

The panel that is about to expand or collapse.

expanding

`YES` specifies that the panel is expanding; `NO` specifies that it is collapsing.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSSavePanel.h`

panelSelectionDidChange:

Tells the delegate that the user changed the selection in the specified Save panel.

```
- (void)panelSelectionDidChange:(id)sender
```

Parameters

sender

The panel whose selection changed.

Availability

Available in Mac OS X v10.3 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSSavePanel.h`

NSOutlineViewDataSource Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSOutlineView.h
Companion guide	Outline View Programming Topics

Overview

`NSOutlineView` objects support a data source delegate in addition to the regular delegate object. The `NSOutlineViewDataSource` protocol defines methods that the outline view invokes as necessary to retrieve data and information about the data from the data source delegate, and—optionally—to update data values.

All the methods in the `NSOutlineViewDataSource` protocol are marked as `@optional`. While this is true, there are cases were you must implement some methods to achieve required functionality, specifically when working with conventional data sources rather that data that is provided by Cocoa bindings.

Required and Optional Methods Using Programmatic Conventions and Cocoa Bindings

If you are using conventional data sources for content you must implement the basic methods that provide the outline view with data: `outlineView:child:ofItem:` (page 3743), `outlineView:isItemExpandable:` (page 3744), `outlineView:numberOfChildrenOfItem:` (page 3746), and `outlineView:objectValueForTableColumn:byItem:` (page 3747). Applications that acquire their data using Cocoa bindings do not need to implement these methods.

Similarly, when using conventional data sources , if you want to allow the user to edit values, you must implement `outlineView:setObjectValue:forTableColumn:byItem:` (page 3748). When these methods are invoked by the outline view, `nil` as the *`item`* refers to the "root" item. `NSOutlineView` requires that each item in the outline view be unique. In order for the collapsed state of an outline view to remain consistent between reloads you must always return the same object for an item. When using Cocoa bindings to provide outline view content, there is no requirement to implement this method.

Note: Some of the methods in this `protocol`, such as `outlineView:child:ofItem:` (page 3743) and `outlineView:numberOfChildrenOfItem:` (page 3746) along with other methods that return data, are called very frequently, so they must be efficient.

Tasks

Working with Items in a View

- `outlineView:child:ofItem:` (page 3743)
  Returns the child item at the specified index of a given item.
- `outlineView:isItemExpandable:` (page 3744)
  Returns a Boolean value that indicates whether the a given item is expandable.
- `outlineView:numberOfChildrenOfItem:` (page 3746)
  Returns the number of child items encompassed by a given item.
- `outlineView:objectValueForTableColumn:byItem:` (page 3747)
  Invoked by *`outlineView`* to return the data object associated with the specified *`item`*.
- `outlineView:setObjectValue:forTableColumn:byItem:` (page 3748)
  Set the data object for a given item in a given column.

Supporting Drag and Drop

- `outlineView:acceptDrop:item:childIndex:` (page 3743)
  Returns a Boolean value that indicates whether a drop operation was successful.
- `outlineView:validateDrop:proposedItem:proposedChildIndex:` (page 3749)
  Used by an outline view to determine a valid drop target.
- `outlineView:namesOfPromisedFilesDroppedAtDestination:forDraggedItems:` (page 3745)
  Returns an array of filenames for the created files that the receiver promises to create.

Supporting Object Persistence

- `outlineView:itemForPersistentObject:` (page 3745)
  Invoked by *`outlineView`* to return the item for the archived *`object`*.
- `outlineView:persistentObjectForItem:` (page 3747)
  Invoked by *`outlineView`* to return an archived object for *`item`*.

Working with a Pasteboard

- `outlineView:writeItems:toPasteboard:` (page 3750)
  Returns a Boolean value that indicates whether a drag operation is allowed.

Sorting

`- outlineView:sortDescriptorsDidChange:` (page 3748)
Invoked by an outline view to notify the data source that the descriptors changed and the data may need to be resorted.

Instance Methods

outlineView:acceptDrop:item:childIndex:

Returns a Boolean value that indicates whether a drop operation was successful.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView acceptDrop:(id < NSDraggingInfo
    >)info item:(id)item childIndex:(NSInteger)index
```

Parameters

outlineView
The outline view that sent the message. *outlineView* must have previously allowed a drop.

info
An object that contains more information about this dragging operation.

item
The parent of the item over which the cursor was placed when the mouse button was released.

index
The index of the child of *item* over which the cursor was placed when the mouse button was released.

Return Value
`YES` if the drop operation was successful, otherwise `NO`.

Discussion
The data source should incorporate the data from the dragging pasteboard in the implementation of this method. You can get the data for the drop operation from info using the `draggingPasteboard` (page 3662) method.

The return value indicates success or failure of the drag operation to the system.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- shouldCollapseAutoExpandedItemsForDeposited:` (page 1843) (NSOutlineView)

Declared In
`NSOutlineView.h`

outlineView:child:ofItem:

Returns the child item at the specified index of a given item.

```
- (id)outlineView:(NSOutlineView *)outlineView child:(NSInteger)index ofItem:(id)item
```

Parameters

outlineView

The outline view that sent the message.

index

The index of the child item from *item* to return.

item

An item in the data source.

Return Value

The child item at `index` of a *item*. If *item* is `nil`, returns the appropriate child item of the root object.

Discussion

Children of a given parent *item* are accessed sequentially. In order for the collapsed state of the outline view to remain consistent when it is reloaded you must always return the same object for a specified *child* and *item*.

Important: While this method is marked as `@optional` in the protocol, **you must implement this method if you are not providing the data for the outline view using Cocoa bindings.**

Special Considerations

`outlineView:child:ofItem:` (page 3743) is called very frequently, so it must be efficient.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

`- outlineView:numberOfChildrenOfItem:` (page 3746)

Declared In

`NSOutlineView.h`

outlineView:isItemExpandable:

Returns a Boolean value that indicates whether the a given item is expandable.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView isItemExpandable:(id)item
```

Parameters

outlineView

The outline view that sent the message.

item

An item in the data source.

Return Value

`YES` if *item* can be expanded to display its children, otherwise `NO`.

Discussion

This method may be called quite often and should be efficient.

Important: While this method is marked as `@optional` in the protocol, **you must implement this method if you are not providing the data for the outline view using Cocoa bindings.**

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSOutlineView.h`

outlineView:itemForPersistentObject:

Invoked by *outlineView* to return the item for the archived *object*.

```
- (id)outlineView:(NSOutlineView *)outlineView itemForPersistentObject:(id)object
```

Parameters

outlineView
The outline view that sent the message.

object
An archived representation of an item in *outlineView*'s data source.

Return Value
The unarchived item corresponding to *object*. If the item is an archived object, this method may return the object.

Discussion
When the outline view is restoring the saved expanded items, this method is called for each expanded item, to translate the archived object to an outline view item.

Special Considerations

You must implement this method if you are automatically saving expanded items (that is, if `autosaveExpandedItems` (page 1830) returns `YES`).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSOutlineView.h`

outlineView:namesOfPromisedFilesDroppedAtDestination:forDraggedItems:

Returns an array of filenames for the created files that the receiver promises to create.

```
- (NSArray *)outlineView:(NSOutlineView *)outlineView
    namesOfPromisedFilesDroppedAtDestination:(NSURL *)dropDestination
    forDraggedItems:(NSArray *)items
```

Parameters

outlineView

The outline view that sent the message.

dropDestination

The drop location where the files are created.

items

The items being dragged.

Return Value

An array of filenames (not full paths) for the created files that the receiver promises to create.

Discussion

For more information on file promise dragging, see documentation on the `NSDraggingSource` protocol and `namesOfPromisedFilesDroppedAtDestination:` (page 3663).

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSOutlineView.h`

outlineView:numberOfChildrenOfItem:

Returns the number of child items encompassed by a given item.

```
- (NSInteger)outlineView:(NSOutlineView *)outlineView numberOfChildrenOfItem:(id)item
```

Parameters

outlineView

The outline view that sent the message.

item

An item in the data source.

Return Value

The number of child items encompassed by *item*. If *item* is `nil`, this method should return the number of children for the top-level item.

Discussion

`outlineView:numberOfChildrenOfItem:` (page 3746) is called very frequently, so it must be efficient.

Important: While this method is marked as `@optional` in the protocol, **you must implement this method if you are not providing the data for the outline view using Cocoa bindings.**

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSOutlineView.h`

outlineView:objectValueForTableColumn:byItem:

Invoked by *outlineView* to return the data object associated with the specified *item*.

```
- (id)outlineView:(NSOutlineView *)outlineView
    objectValueForTableColumn:(NSTableColumn *)tableColumn byItem:(id)item
```

Parameters

outlineView

The outline view that sent the message.

tableColumn

A column in *outlineView*.

item

An item in the data source in the specified *tableColumn* of the view.

Discussion

The item is located in the specified *tableColumn* of the view.

Important: While this method is marked as `@optional` in the protocol, **you must implement this method if you are not providing the data for the outline view using Cocoa bindings.**

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSOutlineView.h`

outlineView:persistentObjectForItem:

Invoked by *outlineView* to return an archived object for *item*.

```
- (id)outlineView:(NSOutlineView *)outlineView persistentObjectForItem:(id)item
```

Parameters

outlineView

The outline view that sent the message.

item

The item for which to return an archived object.

Return Value

An archived representation of *item*. If the item is an archived object, this method may return the item.

Discussion

When the outline view is saving the expanded items, this method is called for each expanded item, to translate the outline view item to an archived object.

Special Considerations

You must implement this method if you are automatically saving expanded items (that is, if `autosaveExpandedItems` (page 1830) returns `YES`).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:setObjectValue:forTableColumn:byItem:

Set the data object for a given item in a given column.

```
- (void)outlineView:(NSOutlineView *)outlineView setObjectValue:(id)object
    forTableColumn:(NSTableColumn *)tableColumn byItem:(id)item
```

Parameters

outlineView
The outline view that sent the message.

object
The new value for the item.

tableColumn
A column in *outlineView*.

item
An item in the data source in the specified *tableColumn* of the view.

Discussion
The item is located in the specified *tableColumn* of the view.

Important: While this method is marked as `@optional` in the protocol, **you must implement this method if you are not providing the data for the outline view using Cocoa bindings.**

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:sortDescriptorsDidChange:

Invoked by an outline view to notify the data source that the descriptors changed and the data may need to be resorted.

```
- (void)outlineView:(NSOutlineView *)outlineView sortDescriptorsDidChange:(NSArray
    *)oldDescriptors
```

Parameters

outlineView
The outline view that sent the message.

oldDescriptors
An array that contains the previous descriptors.

Discussion
The data source typically sorts and reloads the data, and adjusts the selections accordingly. If you need to know the current sort descriptors and the data source does not itself manage them, you can get *outlineView*'s current sort descriptors by sending it a `sortDescriptors` (page 2670) message.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSOutlineView.h`

outlineView:validateDrop:proposedItem:proposedChildIndex:

Used by an outline view to determine a valid drop target.

```
- (NSDragOperation)outlineView:(NSOutlineView *)outlineView validateDrop:(id <
    NSDraggingInfo >)info proposedItem:(id)item proposedChildIndex:(NSInteger)index
```

Parameters
outlineView
The outline view that sent the message.

info
An object that contains more information about this dragging operation.

item
The proposed parent.

index
The proposed child location.

Return Value
A value that indicates which dragging operation the data source will perform.

Discussion
Based on the mouse position, the outline view will suggest a proposed drop location. The data source may "retarget" a drop if desired by calling `setDropItem:dropChildIndex:` (page 1841) and returning something other than `NSDragOperationNone`. You may choose to retarget for various reasons (for example, for better visual feedback when inserting into a sorted position).

Implementation of this method is optional.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSOutlineView.h`

outlineView:writeItems:toPasteboard:

Returns a Boolean value that indicates whether a drag operation is allowed.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView writeItems:(NSArray *)items
    toPasteboard:(NSPasteboard *)pboard
```

Parameters

outlineView

The outline view that invoked the method.

items

An array of the items participating in the drag.

pboard

The pasteboard to which to write the drag data.

Return Value

`YES` if the drag operation is allowed, otherwise `NO`.

Discussion

Invoked by *outlineView* after it has been determined that a drag should begin, but before the drag has been started.

To refuse the drag, return `NO`. To start a drag, return `YES` and place the drag data onto the *pboard* (data, owner, and so on). The drag image and other drag-related information will be set up and provided by the outline view once this call returns with `YES`.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSOutlineView.h`

NSOutlineViewDelegate Protocol Reference

Conforms to	NSControlTextEditingDelegate
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSOutlineView.h
Companion guides	Outline View Programming Topics Drag and Drop Programming Topics for Cocoa

Overview

The `NSOutlineViewDelegate` protocol defines the optional methods implemented by delegates of `NSOutlineView` objects.

Some delegate methods have not yet migrated to the `NSOutlineViewDelegate` protocol, including:

- `outlineViewColumnDidMove:` (page 1844)
- `outlineViewColumnDidResize:` (page 1844)
- `outlineViewItemDidCollapse:` (page 1844)
- `outlineViewItemDidExpand:` (page 1845)
- `outlineViewItemWillCollapse:` (page 1845)
- `outlineViewItemWillExpand:` (page 1845)
- `outlineViewSelectionDidChange:` (page 1846)
- `outlineViewSelectionIsChanging:` (page 1846)

Tasks

Expanding and Collapsing the Outline

`- outlineView:shouldExpandItem:` (page 3759)
Returns a Boolean value that indicates whether the outline view should expand a given item.

- `outlineView:shouldCollapseItem:` (page 3757)
  Returns a Boolean value that indicates whether the outline view should collapse a given item.

Supporting Type Select

- `outlineView:typeSelectStringForTableColumn:item:` (page 3764)
  Returns the string that is used for type selection for a given column and item.
- `outlineView:nextTypeSelectMatchFromItem:toItem:forString:` (page 3756)
  Returns the first item that matches the searchString from within the range of startItem to endItem
- `outlineView:shouldTypeSelectForEvent:withCurrentSearchString:` (page 3762)
  Returns a Boolean value that indicates whether type select should proceed for a given event and search string.

Working with Tooltips

- `outlineView:toolTipForCell:rect:tableColumn:item:mouseLocation:` (page 3763)
  When the cursor pauses over a given cell, the value returned from this method is displayed in a tooltip.

Handling Selection

- `outlineView:shouldSelectTableColumn:` (page 3760)
  Returns a Boolean value that indicates whether the outline view should select a given table column.
- `outlineView:shouldSelectItem:` (page 3760)
  Returns a Boolean value that indicates whether the outline view should select a given item.
- `outlineView:selectionIndexesForProposedSelection:` (page 3757)
  Invoked to allow the delegate to modify the proposed selection.
- `selectionShouldChangeInOutlineView:` (page 3765)
  Returns a Boolean value that indicates whether the outline view should change its selection.

Displaying Cells

- `outlineView:willDisplayCell:forTableColumn:item:` (page 3765)
  Informs the delegate that *`outlineView`* is about to display the cell specified by *`tableColumn`* and *`item`*.
- `outlineView:willDisplayOutlineCell:forTableColumn:item:` (page 3765)
  Informs the delegate that an outline view is about to display a cell used to draw the expansion symbol.
- `outlineView:dataCellForTableColumn:item:` (page 3754)
  Returns the cell to use in a given column for a given item.
- `outlineView:shouldShowOutlineCellForItem:` (page 3761)
  Returns a whether the specified item should display the outline cell (the disclosure triangle).
- `outlineView:shouldShowCellExpansionForTableColumn:item:` (page 3760)
  Invoked to allow the delegate to control cell expansion for a specific column and item.

Moving and Resizing Columns

- `outlineView:shouldReorderColumn:toColumn:` (page 3759)
  Sent to the delegate to allow or prohibit the specified column to be dragged to a new location.

Editing Items

- `outlineView:shouldEditTableColumn:item:` (page 3758)
  Returns a Boolean value that indicates whether the outline view should allow editing of a given item in a given table column.

Working with Table Columns

- `outlineView:mouseDownInHeaderOfTableColumn:` (page 3756)
  Sent to the delegate whenever the mouse button is clicked in *outlineView* while the cursor is in a column header *tableColumn*.
- `outlineView:didClickTableColumn:` (page 3754)
  Sent at the time the mouse button subsequently goes up in *outlineView* and *tableColumn* has been "clicked" without having been dragged anywhere.
- `outlineView:didDragTableColumn:` (page 3755)
  Sent at the time the mouse button goes up in *outlineView* and *tableColumn* has been dragged during the time the mouse button was down.

Customizing Column and Row Sizes

- `outlineView:heightOfRowByItem:` (page 3755)
  Returns the height in points of the row containing *item*.
- `outlineView:sizeToFitWidthOfColumn:` (page 3763)
  Invoked to allow the delegate to provide custom sizing behavior when a column's resize divider is double clicked.

Customizing Tracking Support

- `outlineView:shouldTrackCell:forTableColumn:item:` (page 3762)
  Returns a Boolean value that indicates whether a given cell should be tracked.

Grouping Rows

- `outlineView:isGroupItem:` (page 3755)
  Returns a Boolean that indicates whether a given row should be drawn in the "group row" style.

Instance Methods

outlineView:dataCellForTableColumn:item:

Returns the cell to use in a given column for a given item.

```
- (NSCell *)outlineView:(NSOutlineView *)outlineView
    dataCellForTableColumn:(NSTableColumn *)tableColumn item:(id)item
```

Parameters

outlineView

The outline view that sent the message.

tableColumn

The table column for which the cell is required. This value may be `nil`.

item

The item for which the cell is required.

Return Value

The cell to use in column *tableColumn* for item *item*, or `nil` (see Discussion). The cell must properly implement `copyWithZone:` (since it may be copied by by the outline view).

Discussion

You can return a different data cell for any particular combination of table column and item, or a cell that will be used for the entire row (a full width cell). If *tableColumn* is non-`nil`, you should return a cell. Typically you should default to returning the result from `[tableColumn dataCellForRow:row]`.

When each row (identified by the item) is being drawn, this method is first called with a `nil` value for *tableColumn*. At this time, you can return a cell that will be used to draw the entire row, acting like a group. If you do return a cell for the 'nil' table column, your implementations of the other corresponding datasource and delegate methods must be prepared to be invoked with a `nil` value for *tableColumn*. If do not return a cell for the 'nil' table column, the method will be called once for each column in the outline view, as usual.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSOutlineView.h`

outlineView:didClickTableColumn:

Sent at the time the mouse button subsequently goes up in *outlineView* and *tableColumn* has been "clicked" without having been dragged anywhere.

```
- (void)outlineView:(NSOutlineView *)outlineView didClickTableColumn:(NSTableColumn
    *)tableColumn
```

Availability

Available in Mac OS X v10.3 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:didDragTableColumn:

Sent at the time the mouse button goes up in *outlineView* and *tableColumn* has been dragged during the time the mouse button was down.

```
- (void)outlineView:(NSOutlineView *)outlineView didDragTableColumn:(NSTableColumn
    *)tableColumn
```

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:heightOfRowByItem:

Returns the height in points of the row containing *item*.

```
- (CGFloat)outlineView:(NSOutlineView *)outlineView heightOfRowByItem:(id)item
```

Discussion
Values returned by this method should not include intercell spacing and must be greater than 0. Implement this method to support an outline view with varying row heights.

Special Considerations

For large tables in particular, you should make sure that this method is efficient. NSTableView may cache the values this method returns, so if you would like to change a row's height make sure to invalidate the row height by calling noteHeightOfRowsWithIndexesChanged: (page 2640). NSTableView automatically invalidates its entire row height cache in reloadData (page 2645) and noteNumberOfRowsChanged (page 2641).

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:isGroupItem:

Returns a Boolean that indicates whether a given row should be drawn in the “group row” style.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView isGroupItem:(id)item
```

Parameters

outlineView
The outline view that sent the message.

item
An item in the outline view.

Return Value
YES to indicate a particular row should have the "group row" style drawn for that row, otherwise NO.

Discussion
If the cell in that row is an instance of NSTextFieldCell and contains only a string value, the “group row” style attributes are automatically applied for that cell.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:mouseDownInHeaderOfTableColumn:

Sent to the delegate whenever the mouse button is clicked in *outlineView* while the cursor is in a column header *tableColumn*.

```
- (void)outlineView:(NSOutlineView *)outlineView
    mouseDownInHeaderOfTableColumn:(NSTableColumn *)tableColumn
```

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:nextTypeSelectMatchFromItem:toItem:forString:

Returns the first item that matches the searchString from within the range of startItem to endItem

```
- (id)outlineView:(NSOutlineView *)outlineView
    nextTypeSelectMatchFromItem:(id)startItem toItem:(id)endItem forString:(NSString
     *)searchString
```

Parameters
outlineView
The outline view that sent the message.

startItem
The first item to search.

endItem
The item before which to stop searching. It is possible for endItem to be less than startItem if the search will wrap.

searchString
The string for which to search.

Return Value
The first item—from within the range of *startItem* to *endItem*—that matches the *searchString*, or `nil` if there is no match.

Discussion
Implement this method if you want to control how type selection works. You should include *startItem* as a possible match, but do not include *endItem*.

It is not necessary to implement this method in order to support type select.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSOutlineView.h`

outlineView:selectionIndexesForProposedSelection:

Invoked to allow the delegate to modify the proposed selection.

```
- (NSIndexSet *)outlineView:(NSOutlineView *)outlineView
    selectionIndexesForProposedSelection:(NSIndexSet *)proposedSelectionIndexes
```

Parameters
outlineView
The outline view that sent the message.
proposedSelectionIndexes
An index set containing the indexes of the proposed selection.

Return Value
An `NSIndexSet` instance containing the indexes of the new selection. Return *proposedSelectionIndexes* if the proposed selection is acceptable, or the value of the table view's existing selection to avoid changing the selection.

Discussion
This method may be called multiple times with one new index added to the existing selection to find out if a particular index can be selected when the user is extending the selection with the keyboard or mouse.

Implementation of this method is optional. If implemented, this method will be called instead of `outlineView:willDisplayOutlineCell:forTableColumn:item:` (page 3765).

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSOutlineView.h`

outlineView:shouldCollapseItem:

Returns a Boolean value that indicates whether the outline view should collapse a given item.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView shouldCollapseItem:(id)item
```

Parameters

outlineView

The outline view that sent the message.

item

The item that should collapse.

Return Value

YES to permit *outlineView* to collapse *item*, NO to deny permission.

Discussion

The delegate can implement this method to disallow collapsing of specific items. For example, if the first row of your outline view should not be collapsed, your delegate method could contain this line of code:

```
return [outlineView rowForItem:item]!=0;
```

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSOutlineView.h

outlineView:shouldEditTableColumn:item:

Returns a Boolean value that indicates whether the outline view should allow editing of a given item in a given table column.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView shouldEditTableColumn:(NSTableColumn
    *)tableColumn item:(id)item
```

Return Value

YES to permit *outlineView* to edit the cell specified by *tableColumn* and *item*, NO to deny permission.

If this method returns YES, the cell may still not be editable—for example, if you have set up a custom NSTextFieldCell as a data cell, it must return YES for isEditable to allow editing.

Discussion

The delegate can implement this method to disallow editing of specific cells.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- outlineView:setObjectValue:forTableColumn:byItem:

Declared In

NSOutlineView.h

outlineView:shouldExpandItem:

Returns a Boolean value that indicates whether the outline view should expand a given item.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView shouldExpandItem:(id)item
```

Parameters

outlineView

The outline view that sent the message.

item

The item that should expand.

Return Value

YES to permit *outlineView* to expand *item*, NO to deny permission.

Discussion

The delegate can implement this method to disallow expanding of specific items.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSOutlineView.h

outlineView:shouldReorderColumn:toColumn:

Sent to the delegate to allow or prohibit the specified column to be dragged to a new location.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView
    shouldReorderColumn:(NSInteger)columnIndex toColumn:(NSInteger)newColumnIndex
```

Parameters

outlineView

The outline view that sent the message.

columnIndex

The index of the column being dragged.

newColumnIndex

The proposed target index of the column.

Return Value

YES if the column reordering should be allowed, otherwise NO.

Discussion

When a column is initially dragged by the user, the delegate is first called with a *newColumnIndex* value of -1. Returning NO will disallow that column from being reordered at all. Returning YES allows it to be reordered, and the delegate will be called again when the column reaches a new location.

The actual NSTableColumn instance can be retrieved from the tableColumns (page 2670) array.

If this method is not implemented, all columns are considered reorderable.

Availability

Available in Mac OS X v10.6 and later.

Declared In
NSOutlineView.h

outlineView:shouldSelectItem:

Returns a Boolean value that indicates whether the outline view should select a given item.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView shouldSelectItem:(id)item
```

Return Value
YES to permit *outlineView* to select *item*, NO to deny permission.

Discussion
The delegate can implement this method to disallow selection of particular items.

For better performance and finer grain control over the selection, use `outlineView:dataCellForTableColumn:item:` (page 3754).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:shouldSelectTableColumn:

Returns a Boolean value that indicates whether the outline view should select a given table column.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView
    shouldSelectTableColumn:(NSTableColumn *)tableColumn
```

Return Value
YES to permit *outlineView* to select *tableColumn*, NO to deny permission.

Discussion
The delegate can implement this method to disallow selection of specific columns.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:shouldShowCellExpansionForTableColumn:item:

Invoked to allow the delegate to control cell expansion for a specific column and item.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView
    shouldShowCellExpansionForTableColumn:(NSTableColumn *)tableColumn item:(id)item
```

Parameters

outlineView
The outline view that sent the message.

tableColumn
A table column in the outline view.

item
An item in the outline view.

Return Value
YES to allow an expansion tooltip to appear in the column *tableColumn* for item *item*, otherwise NO.

Discussion
Cell expansion can occur when the mouse hovers over the specified cell and the cell contents are unable to be fully displayed within the cell. If this method returns YES, the full cell contents will be shown in a special floating tool tip view, otherwise the content is truncated.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:shouldShowOutlineCellForItem:

Returns a whether the specified item should display the outline cell (the disclosure triangle).

```
- (BOOL)outlineView:(NSOutlineView *)outlineView
    shouldShowOutlineCellForItem:(id)item
```

Parameters

outlineView
The outline view that sent the message.

item
An item in the outline view.

Return Value
YES if the outline cell should be displayed, otherwise NO.

Discussion
Returning NO causes frameOfOutlineCellAtRow: (page 1833) to return NSZeroRect, hiding the cell. In addition, the row will not be collapsable by keyboard shortcuts.

This method will only be called for expandable rows.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSOutlineView.h

outlineView:shouldTrackCell:forTableColumn:item:

Returns a Boolean value that indicates whether a given cell should be tracked.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView shouldTrackCell:(NSCell *)cell
    forTableColumn:(NSTableColumn *)tableColumn item:(id)item
```

Parameters

outlineView

The outline view that sent the message.

cell

The cell used to display item *item* in column *tableColumn*

tableColumn

A table column in the outline view.

item

An item in the outline view.

Return Value

YES if the cell should be tracked for the item *item* in column *tableColumn*, otherwise NO.

Discussion

Normally, only selectable or selected cells can be tracked. If you implement this method, cells which are not selectable or selected can be tracked (and vice-versa). For example, this allows you to have a button cell in a table which does not change the selection, but can still be clicked on and tracked.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSOutlineView.h

outlineView:shouldTypeSelectForEvent:withCurrentSearchString:

Returns a Boolean value that indicates whether type select should proceed for a given event and search string.

```
- (BOOL)outlineView:(NSOutlineView *)outlineView shouldTypeSelectForEvent:(NSEvent
    *)event withCurrentSearchString:(NSString *)searchString
```

Parameters

outlineView

The outline view that sent the message.

event

The event that caused the message to be sent.

searchString

The string for which searching is to proceed. The search string is nil if no type select has begun.

Return Value

YES if type select should proceed, otherwise NO.

Discussion

Generally, this method will be called from keyDown: (page 2159) and the event will be a key event.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:sizeToFitWidthOfColumn:

Invoked to allow the delegate to provide custom sizing behavior when a column's resize divider is double clicked.

```
- (CGFloat)outlineView:(NSOutlineView *)outlineView
    sizeToFitWidthOfColumn:(NSInteger)column
```

Parameters

outlineView
The outline view that sent the message.

column
The index of the column.

Return Value
The width of the specified column.

Discussion
By default, NSOutlineView iterates every row in the table, accesses a cell via preparedCellAtColumn:row: (page 2643), and requests the cellSize (page 549) to find the appropriate largest width to use.

For accurate results and performance, it is recommended that this method is implemented when using large tables. By default, large tables use a monte carlo simulation instead of interating every row.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSOutlineView.h

outlineView:toolTipForCell:rect:tableColumn:item:mouseLocation:

When the cursor pauses over a given cell, the value returned from this method is displayed in a tooltip.

```
- (NSString *)outlineView:(NSOutlineView *)outlineView toolTipForCell:(NSCell *)cell
    rect:(NSRectPointer)rect tableColumn:(NSTableColumn *)tc item:(id)item
   mouseLocation:(NSPoint)mouseLocation
```

Parameters

outlineView
The outline view that sent the message.

cell
The cell for which to generate a tooltip.

rect
The proposed active area of the tooltip. To control the default active area, you can modify the *rect* parameter. By default, *rect* is computed as `[cell drawingRectForBounds:cellFrame]`.

tc
The table column that contains *cell*.

item
The item for which to display a tooltip.

mouseLocation
The current mouse location in view coordinates.

Return Value
If you don't want a tooltip at that location, return `nil` or the empty string.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSOutlineView.h`

outlineView:typeSelectStringForTableColumn:item:

Returns the string that is used for type selection for a given column and item.

```
- (NSString *)outlineView:(NSOutlineView *)outlineView
    typeSelectStringForTableColumn:(NSTableColumn *)tableColumn item:(id)item
```

Parameters
outlineView
The outline view that sent the message.

tableColumn
A table column in the outline view.

item
An item in the outline view.

Return Value
The string that is used for type selection. You may want to change what is searched for based on what is displayed, or simply return nil for that row and/or column to not be searched

Discussion
Implement this method if you want to control the string that is used for type selection. You may want to change what is searched for based on what is displayed, or simply return `nil` to specify that the given row and/or column should not be searched. By default, all cells with text in them are searched.

The default value when this delegate method is not implemented is:

```
[[outlineView preparedCellAtColumn:tableColumn row:[outlineView rowForItem:item]]
 stringValue]
```

and you can return this value from the delegate method if you wish.

Availability
Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:willDisplayCell:forTableColumn:item:

Informs the delegate that *outlineView* is about to display the cell specified by *tableColumn* and *item*.

```
- (void)outlineView:(NSOutlineView *)outlineView willDisplayCell:(id)cell
    forTableColumn:(NSTableColumn *)tableColumn item:(id)item
```

Discussion
The delegate can modify *cell* to alter its display attributes; for example, making uneditable values display in italic or gray text.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

outlineView:willDisplayOutlineCell:forTableColumn:item:

Informs the delegate that an outline view is about to display a cell used to draw the expansion symbol.

```
- (void)outlineView:(NSOutlineView *)outlineView willDisplayOutlineCell:(id)cell
    forTableColumn:(NSTableColumn *)tableColumn item:(id)item
```

Discussion
Informs the delegate that *outlineView* is about to display *cell*—an expandable cell (a cell that has the expansion symbol)—for the column and item specified by *tableColumn* and *item*. The delegate can modify cell to alter its display attributes.

This method is not invoked when *outlineView* is about to display a non-expandable cell.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSOutlineView.h

selectionShouldChangeInOutlineView:

Returns a Boolean value that indicates whether the outline view should change its selection.

```
- (BOOL)selectionShouldChangeInOutlineView:(NSOutlineView *)outlineView
```

Return Value
YES to permit *outlineView* to change its selection (typically a row being edited), NO to deny permission.

Discussion
For example, if the user is editing a cell and enters an improper value, the delegate can prevent the user from selecting or editing any other cells until a proper value has been entered into the original cell. The delegate can implement this method for complex validation of edited rows based on the values of any of their cells.

Availability
Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSOutlineView.h`

NSPasteboardItemDataProvider Protocol Reference

Conforms to	NSObject
Availability	Available in Mac OS X v10.6 and later.
Declared in	NSPasteboard.h
Companion guides	Pasteboard Programming Guide Drag and Drop Programming Topics for Cocoa Services Implementation Guide

Overview

This protocol is implemented by the data provider of a pasteboard item to provide the data for a particular UTI type.

You can specify an object as a pasteboard data provider for a pasteboard item using `NSPasteboardItem`'s `setDataProvider:forTypes:` (page 1914) method. The data provider must implement this protocol to provide data upon request.

Tasks

Providing Data

- `pasteboard:item:provideDataForType:` (page 3767) *required method*
  Asks the receiver to provide data for a specified type to a given pasteboard. (required)
- `pasteboardFinishedWithDataProvider:` (page 3768)
  Informs the receiver that the pasteboard no longer needs the data provider for any of its pasteboard items.

Instance Methods

pasteboard:item:provideDataForType:

Asks the receiver to provide data for a specified type to a given pasteboard. (required)

```
- (void)pasteboard:(NSPasteboard *)pasteboard
    item:(NSPasteboardItem *)item
    provideDataForType:(NSString *)type
```

Parameters

pasteboard

A pasteboard to which the receiver has promised to provide data.

item

A pasteboard item for which the receiver has promised to provide data

type

A UTI type string.

Discussion

The receiver was previously set as the provider using `setDataProvider:forTypes:` (page 1914).

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSPasteboardItem.h`

pasteboardFinishedWithDataProvider:

Informs the receiver that the pasteboard no longer needs the data provider for any of its pasteboard items.

```
- (void)pasteboardFinishedWithDataProvider:(NSPasteboard *)pasteboard
```

Parameters

pasteboard

A pasteboard.

Discussion

One data provider can provide data for more than one pasteboard item. This method is called when the pasteboard no longer needs the data provider for any of its pasteboard items. This can be either because the data provider has fulfilled all promises, or because ownership of the pasteboard has changed.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSPasteboardItem.h`

NSPasteboardReading Protocol Reference

Conforms to	NSObject
Availability	Available in Mac OS X v10.6 and later.
Declared in	NSPasteboard.h
Companion guides	Pasteboard Programming Guide Drag and Drop Programming Topics for Cocoa Services Implementation Guide
Related sample code	DemoMonkey

Overview

The `NSPasteboardReading` protocol specifies the interface for initializing an object from a pasteboard. The Cocoa framework classes `NSString`, `NSAttributedString`, `NSURL`, `NSColor`, `NSSound`, `NSImage`, and `NSPasteboardItem` implement this protocol. You can make your custom class conform to this protocol so that you can read instances from a pasteboard using the `readObjectsForClasses:options:` (page 1897) method of `NSPasteboard`.

Tasks

Required Methods

`+ readableTypesForPasteboard:` (page 3770) *required method*

Returns an array of UTI strings of data types the receiver can read from the pasteboard and be initialized from. (required)

`+ readingOptionsForType:pasteboard:` (page 3770) *required method*

Returns options for reading data of a specified type from a given pasteboard. (required)

Optional Method

`- initWithPasteboardPropertyList:ofType:` (page 3771)

Initializes an instance with a property list object and a type string.

Class Methods

readableTypesForPasteboard:

Returns an array of UTI strings of data types the receiver can read from the pasteboard and be initialized from. (required)

```
+ (NSArray *)readableTypesForPasteboard:(NSPasteboard *)pasteboard
```

Parameters

pasteboard

A pasteboard.

You can use the pasteboard argument to provide different types based on the pasteboard name, should you need to do so.

Return Value

An array of UTI strings of data types instances of the receiver can read from the pasteboard and be initialized from.

Discussion

By default, the data for a type is provided to `initWithPasteboardPropertyList:ofType:` (page 3771) as an instance of `NSData`. If you implement `readingOptionsForType:pasteboard:` (page 3770) and specify a different option, the `NSData` object for a type can be converted to an `NSString` object or any other property list object.

Special Considerations

Do not perform other pasteboard operations in the method implementation.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSPasteboard.h`

readingOptionsForType:pasteboard:

Returns options for reading data of a specified type from a given pasteboard. (required)

```
+ (NSPasteboardReadingOptions)readingOptionsForType:(NSString *)type
    pasteboard:(NSPasteboard *)pasteboard
```

Parameters

type

A UTI supported by instances of the receiver for reading (one of the types returned by `readableTypesForPasteboard:` (page 3770)).

pasteboard

A pasteboard.

You can use the pasteboard argument to provide return different based on the pasteboard name, should you need to do so.

Return Value
Options for reading data of *type* from *pasteboard*. For a list of valid values, see "Pasteboard Reading Options" (page 3772).

Special Considerations

Do not perform other pasteboard operations in the method implementation.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSPasteboard.h

Instance Methods

initWithPasteboardPropertyList:ofType:

Initializes an instance with a property list object and a type string.

```
- (id)initWithPasteboardPropertyList:(id)propertyList
    ofType:(NSString *)type
```

Parameters

propertyList

A property list containing data to initialize the receiver.

By default, the property list object is an instance of `NSData`. If you implement `readingOptionsForType:pasteboard:` (page 3770) and specify an option other than `NSPasteboardReadingAsData` (page 3772), the *propertyList* may be any other property list object.

type

A UTI supported by the receiver for reading (one of the types returned by `readableTypesForPasteboard:` (page 3770)).

Return Value
An object initialized using the data in *propertyList*.

Special Considerations

This method is considered optional because, if `readableTypesForPasteboard:` (page 3770) returns just a single type, and that type uses the `NSPasteboardReadingAsKeyedArchive` (page 3772) reading option, then instances are initialized using `initWithCoder:` instead of this method.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSPasteboard.h

Constants

Pasteboard Reading Options

Options to specify how data on the pasteboard should be interpreted to initialize an object in `initWithPasteboardPropertyList:ofType:` (page 3771).

```
enum {
    NSPasteboardReadingAsData         = 0,
    NSPasteboardReadingAsString       = 1 << 0,
    NSPasteboardReadingAsPropertyList = 1 << 1,
    NSPasteboardReadingAsKeyedArchive = 1 << 2
};
```

Constants

`NSPasteboardReadingAsData`

Reads data from the pasteboard as-is and returns it as an `NSData` object.

This is the default value.

Available in Mac OS X v10.6 and later.

Declared in `NSPasteboard.h`.

`NSPasteboardReadingAsString`

Reads data from the pasteboard and converts it to an `NSString` object.

Available in Mac OS X v10.6 and later.

Declared in `NSPasteboard.h`.

`NSPasteboardReadingAsPropertyList`

Reads data from the pasteboard and un-serializes it as a property list.

Available in Mac OS X v10.6 and later.

Declared in `NSPasteboard.h`.

`NSPasteboardReadingAsKeyedArchive`

Reads data from the pasteboard and uses `initWithCoder:` to initialize the object.

Available in Mac OS X v10.6 and later.

Declared in `NSPasteboard.h`.

Discussion

You can specify only one option from this list. If you do not specify an option, the default `NSPasteboardReadingAsData` (page 3772) is used.

NSPasteboardReadingOptions

A type to specify how data on the pasteboard should be interpreted to initialize an object in `initWithPasteboardPropertyList:ofType:` (page 3771).

```
typedef NSUInteger NSPasteboardReadingOptions;
```

Discussion

For valid values, see "Pasteboard Reading Options" (page 3772).

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSPasteboard.h`

NSPasteboardWriting Protocol Reference

Conforms to	NSObject
Availability	Available in Mac OS X v10.6 and later.
Declared in	NSPasteboard.h
Companion guides	Pasteboard Programming Guide Drag and Drop Programming Topics for Cocoa Services Implementation Guide
Related sample code	DemoMonkey

Overview

The `NSPasteboardWriting` protocol specifies the interface for retrieving a representation of an object that can be written to a pasteboard. The Cocoa framework classes `NSString`, `NSAttributedString`, `NSURL`, `NSColor`, `NSSound`, `NSImage`, and `NSPasteboardItem` implement this protocol. You can make your custom class conform to this protocol so that you can write instances of the class to a pasteboard using the `writeObjects:` (page 1903) method of `NSPasteboard`.

Tasks

Required Methods

`- writableTypesForPasteboard:` (page 3776) *required method*

Returns an array of UTI strings of data types the receiver can write to a given pasteboard. (required)

`- pasteboardPropertyListForType:` (page 3776) *required method*

Returns a property list object to represent the receiver on a pasteboard as an object of a specified type. (required)

Optional Method

`- writingOptionsForType:pasteboard:` (page 3777)

Returns options for writing data of a specified type to a given pasteboard.

Instance Methods

pasteboardPropertyListForType:

Returns a property list object to represent the receiver on a pasteboard as an object of a specified type. (required)

```
- (id)pasteboardPropertyListForType:(NSString *)type
```

Parameters

type

One of the types the receiver supports for writing (one of the UTIs returned by its implementation of `writableTypesForPasteboard:` (page 3776)).

Return Value

A property list object to represent the receiver on a pasteboard as an object of type *type*.

Discussion

The returned value will commonly be the `NSData` object for the specified data type. However, if this method returns either a string, or any other property-list type, the pasteboard will automatically convert these items to the correct data format required for the pasteboard.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSPasteboard.h`

writableTypesForPasteboard:

Returns an array of UTI strings of data types the receiver can write to a given pasteboard. (required)

```
- (NSArray *)writableTypesForPasteboard:(NSPasteboard *)pasteboard
```

Parameters

pasteboard

A pasteboard.

You can use this argument to provide different options based on the pasteboard name, if you need to.

Return Value

An array of UTI strings of data types the receiver can write to *pasteboard*.

Discussion

By default, data for the first returned type is put onto the pasteboard immediately, with the remaining types being promised.

To change the default behavior, implement -writingOptionsForType:pasteboard: and return `NSPasteboardWritingPromised` (page 3778) to lazily provide data for types, return no option to provide the data for that type immediately. Use the pasteboard argument to provide different types based on the pasteboard name, if desired. Do not perform other pasteboard operations in the method implementation.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSPasteboard.h`

writingOptionsForType:pasteboard:

Returns options for writing data of a specified type to a given pasteboard.

```
- (NSPasteboardWritingOptions)writingOptionsForType:(NSString *)type
    pasteboard:(NSPasteboard *)pasteboard
```

Parameters

type

One of the types the receiver supports for writing (one of the UTIs returned by its implementation of `writableTypesForPasteboard:` (page 3776)).

pasteboard

A pasteboard.

You can use this argument to provide different options based on the pasteboard name, if you need to.

Return Value
Options for writing data of type `type` to *pasteboard*. Return `0` for no options, or a value given in "Pasteboard Writing Options" (page 3777).

Special Considerations

Do not perform other pasteboard operations in the method implementation.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSPasteboard.h`

Constants

Pasteboard Writing Options

Constant to specify options for writing to a pasteboard, used by `writingOptionsForType:pasteboard:` (page 3777).

```
enum {
    NSPasteboardWritingPromised = 1 << 9,
};
```

Constants

`NSPasteboardWritingPromised`

Data for a type with this option will be promised, not immediately written.

Available in Mac OS X v10.6 and later.

Declared in `NSPasteboard.h`.

NSPasteboardWritingOptions

Type to specify options for writing to a pasteboard.

```
typedef NSUInteger NSPasteboardWritingOptions;
```

Discussion

For possible values, see "Pasteboard Writing Options" (page 3777).

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSPasteboard.h`

NSPathCellDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSPathCell.h

Overview

The `NSPathCellDelegate` optional protocol enables the delegate of an `NSPathCell` object to customize the Open panel or pop-up menu of a path whose style is set to `NSPathStylePopUp` (page 1930).

Tasks

Customizing the Open Panel

`- pathCell:willDisplayOpenPanel:` (page 3779)
Implement this method to customize the Open panel shown by a pop-up–style path.

Customizing the Menu

`- pathCell:willPopUpMenu:` (page 3780)
Implement this method to customize the menu of a pop-up–style path.

Instance Methods

pathCell:willDisplayOpenPanel:

Implement this method to customize the Open panel shown by a pop-up–style path.

```
- (void)pathCell:(NSPathCell *)pathCell
    willDisplayOpenPanel:(NSOpenPanel *)openPanel
```

Parameters

pathCell

The path cell that sent the message.

openPanel

The Open panel to be displayed.

Discussion

This method is called before the Open panel is shown but after its allowed file types are set to the cell's allowed types. At this time, you can further customize the Open panel as required. This method is called only when the style is set to `NSPathStylePopUp`.

Implementation of this method is optional.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSPathCell.h`

pathCell:willPopUpMenu:

Implement this method to customize the menu of a pop-up–style path.

```
- (void)pathCell:(NSPathCell *)pathCell
    willPopUpMenu:(NSMenu *)menu
```

Parameters

pathCell

The path cell that sent the message.

menu

The pop-up menu to be displayed.

Discussion

This method is called before the pop-up menu is shown. At this time, you can further customize the menu as required, adding and removing items. This method is called only when the style is set to `NSPathStylePopUp`.

Implementation of this method is optional.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSPathCell.h`

NSPathControlDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSPathControl.h

Overview

The `NSPathControlDelegate` optional protocol is implemented by the delegate of an `NSPathControl` object to support dragging to and from the control.

Tasks

Dragging Support

- `pathControl:shouldDragPathComponentCell:withPasteboard:` (page 3782)
  Implement this method to enable dragging from the control.
- `pathControl:validateDrop:` (page 3783)
  Implement this method to enable dragging onto the control.
- `pathControl:acceptDrop:` (page 3782)
  Implement this method to accept previously validated contents dropped onto the control.

Customizing a Pop-Up–Style Path

- `pathControl:willDisplayOpenPanel:` (page 3783)
  Implement this method to customize the Open panel shown by a pop-up–style path.
- `pathControl:willPopUpMenu:` (page 3784)
  Implement this method to customize the menu of a pop-up–style path.

Instance Methods

pathControl:acceptDrop:

Implement this method to accept previously validated contents dropped onto the control.

```
- (BOOL)pathControl:(NSPathControl *)pathControl acceptDrop:(id < NSDraggingInfo
    >)info
```

Parameters

pathControl

The path control that sent the message.

info

An object containing details about this dragging operation.

Discussion

In order to accept the dropped contents previously accepted from `pathControl:validateDrop:` (page 3783), you must implement this method. This method is called from `performDragOperation:` (page 3657). You should change the URL value based on the dragged information.

If not implemented, and the control's cell is editable, the drop is accepted if it contains an `NSURLPboardType` or `NSFilenamesPboardType` that conforms to the cell's allowed types. The cell's URL value is automatically changed, and the action is invoked. Implementation of this method is optional.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSPathControl.h`

pathControl:shouldDragPathComponentCell:withPasteboard:

Implement this method to enable dragging from the control.

```
- (BOOL)pathControl:(NSPathControl *)pathControl
    shouldDragPathComponentCell:(NSPathComponentCell *)pathComponentCell
    withPasteboard:(NSPasteboard *)pasteboard
```

Parameters

pathControl

The path control that sent the message.

pathComponentCell

The path component cell from which the drag is beginning.

pasteboard

The pasteboard.

Discussion
This method is called when a drag is about to begin. You can refuse to allow the drag to happen by returning `NO` and allow it by returning `YES`. By default, the pasteboard automatically has the following types on it: `NSStringPboardType`, `NSURLPboardType` (if there is a URL value for the cell being dragged), and `NSFilenamesPboardType` (if the URL value returns `YES` from `-isFileURL`). You can customize the types placed on the pasteboard at this time, if desired. Implementation of this method is optional.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPathControl.h`

pathControl:validateDrop:

Implement this method to enable dragging onto the control.

```
- (NSDragOperation)pathControl:(NSPathControl *)pathControl validateDrop:(id <
    NSDraggingInfo >)info
```

Parameters

pathControl
The path control that sent the message.

info
An object containing details about this dragging operation.

Discussion
This method is called when something is dragged over the control. Return `NSDragOperationNone` to refuse the drop, or return anything else to accept it.

If not implemented, and the control's cell is editable, the drop is accepted if it contains an `NSURLPboardType` or `NSFilenamesPboardType` that conforms to the cell's allowed types. Implementation of this method is optional.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPathControl.h`

pathControl:willDisplayOpenPanel:

Implement this method to customize the Open panel shown by a pop-up–style path.

```
- (void)pathControl:(NSPathControl *)pathControl willDisplayOpenPanel:(NSOpenPanel
    *)openPanel
```

Parameters

pathControl
The path control displaying the Open panel.

openPanel
The Open panel to be displayed.

Discussion
This method is called before the Open panel is shown but after its allowed file types are set to the cell's allowed types. At this time, you can further customize the Open panel as required. This method is called only when the style is set to `NSPathStylePopUp`. Implementation of this method is optional.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPathControl.h`

pathControl:willPopUpMenu:

Implement this method to customize the menu of a pop-up–style path.

```
- (void)pathControl:(NSPathControl *)pathControl willPopUpMenu:(NSMenu *)menu
```

Parameters

pathControl
The path control displaying the pop-up menu.

menu
The pop-up menu to be displayed.

Discussion
This method is called before the pop-up menu is shown. At this time, you can further customize the menu as required, adding and removing items. This method is called only when the style is set to `NSPathStylePopUp`. Implementation of this method is optional.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPathControl.h`

NSPlaceholders Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSKeyValueBinding.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Cocoa Bindings Programming Topics

Overview

The `NSPlaceholders` protocol provides an interface that allows an object to register default placeholders that will be displayed for a binding, when no other placeholder has been specified. Individual placeholder values can be specified for each of the marker objects (described in "`Selection Markers`" (page 3786)), as well as when the property is `nil`.

Placeholders are used when a property of an instance of the receiving class is accessed through a key value coding compliant method, and returns `nil` or a specialized marker.

Tasks

Managing Default Placeholders

`+ setDefaultPlaceholder:forMarker:withBinding:` (page 3786)

Sets *`placeholder`* as the default placeholder for the *`binding`*, when a key value coding compliant property of an instance of the receiving class returns the value specified by *`marker`*, and no other placeholder has been specified.

`+ defaultPlaceholderForMarker:withBinding:` (page 3786)

Returns an object that will be used as the placeholder for the *`binding`*, when a key value coding compliant property of an instance of the receiving class returns the value specified by *`marker`*, and no other placeholder has been specified.

Class Methods

defaultPlaceholderForMarker:withBinding:

Returns an object that will be used as the placeholder for the *binding*, when a key value coding compliant property of an instance of the receiving class returns the value specified by *marker*, and no other placeholder has been specified.

```
+ (id)defaultPlaceholderForMarker:(id)marker withBinding:(NSString *)binding
```

Discussion
The *marker* can be `nil` or one of the constants described in "Selection Markers" (page 3786).

Availability
Available in Mac OS X v10.3 and later.

See Also
`+ setDefaultPlaceholder:forMarker:withBinding:` (page 3786)

Declared In
`NSKeyValueBinding.h`

setDefaultPlaceholder:forMarker:withBinding:

Sets *placeholder* as the default placeholder for the *binding*, when a key value coding compliant property of an instance of the receiving class returns the value specified by *marker*, and no other placeholder has been specified.

```
+ (void)setDefaultPlaceholder:(id)placeholder forMarker:(id)marker
    withBinding:(NSString *)binding
```

Discussion
The *marker* can be `nil` or one of the constants described in "Selection Markers" (page 3786).

Availability
Available in Mac OS X v10.3 and later.

See Also
`+ defaultPlaceholderForMarker:withBinding:` (page 3786)

Declared In
`NSKeyValueBinding.h`

Constants

Selection Markers

The following constants are used to describe special cases for a controller's selection.

```
id NSMultipleValuesMarker;
id NSNoSelectionMarker;
id NSNotApplicableMarker;
```

Constants

`NSMultipleValuesMarker`

This marker indicates that a key's value contains multiple values that differ.

A binding can be configured to always return this marker for multiple items, even if the values are the same.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueBinding.h`.

`NSNoSelectionMarker`

This marker indicates that the controller's selection is currently empty.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueBinding.h`.

`NSNotApplicableMarker`

This marker indicates that an object is not key-value coding compliant for the requested key.

Available in Mac OS X v10.3 and later.

Declared in `NSKeyValueBinding.h`.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`NSKeyValueBinding.h`

NSPrintPanelAccessorizing Protocol Reference

Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSPrintPanel.h

Overview

The `NSPrintPanelAccessorizing` protocol declares two methods that the `NSPrintPanel` class uses to get information from a printing accessory controller.

A printing accessory controller manages a custom print panel accessory view and used to coordinate print settings. If you are implementing a custom printing accessory view, your controller must support this protocol. Implementation of only one method in the protocol is actually required. The other method is considered optional and is used to support the print panel's built-in preview facilities.

Tasks

Responding to Being Loaded from a Nib File

- `localizedSummaryItems` (page 3790) *required method*
  Returns an array of dictionaries containing the localized user setting summary strings. (required)
- `keyPathsForValuesAffectingPreview` (page 3789)
  Returns a set of strings identifying the key paths for any properties that might affect the built-in print preview. (optional)

Instance Methods

keyPathsForValuesAffectingPreview

Returns a set of strings identifying the key paths for any properties that might affect the built-in print preview. (optional)

```
- (NSSet *)keyPathsForValuesAffectingPreview
```

Return Value
A set of `NSString` objects identifying one or more key paths. Only key paths for properties that might affect the contents of the print preview should be returned.

Discussion
If an accessory view modifies printing-related properties that are used by the print preview, you should implement this method to return the key paths for those properties. For example, if you write an accessory view that lets the user change the left and right document margins in the current `NSPrintInfo` object, you would return the following key paths: `representedObject.leftMargin`, `representedObject.rightMargin`. (The `NSPrintInfo` object is the represented object of the accessory controller.)

Implementation of this method is optional. You do not need to implement this method if you are not using the `NSPrintPanel` object's built-in preview facilities. If you do use these facilities, however, you should implement this method.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPrintPanel.h`

localizedSummaryItems

Returns an array of dictionaries containing the localized user setting summary strings. (required)

```
- (NSArray *)localizedSummaryItems
```

Return Value
An array of `NSDictionary` objects, each of which contains a `NSPrintPanelAccessorySummaryItemNameKey` and `NSPrintPanelAccessorySummaryItemDescriptionKey` key. The values for the keys are both strings. This method must not return `nil`.

Discussion
Accessory panels must implement this method to return information about the panel's current settings. The returned array should contain a dictionary for each setting that is managed by the accessory panel and each dictionary should contain two key-value pairs identifying the name of the setting and its current value.

Your accessory view must be KVO-compliant for the `localizedSummaryItems` key path because `NSPrintPanel` object observes that key path and uses it to keep the contents of the summary view up to date. This means your view should manually send KVO notifications to observers for the `localizedSummaryItems` key path whenever the contents of the set of summary items changes. For more information on supporting key-value observing and manual notifications, see *Key-Value Observing Programming Guide*.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSPrintPanel.h`

Constants

Printing Summary Item Keys

These keys must be included in the dictionaries returned by the `localizedSummaryItems` (page 3790) method.

```
NSString *NSPrintPanelAccessorySummaryItemNameKey;
NSString *NSPrintPanelAccessorySummaryItemDescriptionKey;
```

Constants

`NSPrintPanelAccessorySummaryItemNameKey`

Used as a key to specify the name of the accessory panel setting. The corresponding value should be an `NSString` object.

Available in Mac OS X v10.5 and later.

Declared in `NSPrintPanel.h`.

`NSPrintPanelAccessorySummaryItemDescriptionKey`

Used as a key to identify the current value of the accessory panel setting. The corresponding value should be an `NSString` object.

Available in Mac OS X v10.5 and later.

Declared in `NSPrintPanel.h`.

Declared In

`NSPrintPanel.h`

NSRuleEditorDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSRuleEditor.h
Companion guides	Control and Cell Programming Topics for Cocoa Predicate Programming Guide

Overview

The `NSRuleEditorDelegate` protocol defines the optional methods implemented by delegates of `NSRuleEditor` objects.

Tasks

Providing Data

- `ruleEditor:child:forCriterion:withRowType:` (page 3794) *required method*
  Returns the child of a given item at a given index. (required)
- `ruleEditor:displayValueForCriterion:inRow:` (page 3794) *required method*
  Returns the value for a given criterion. (required)
- `ruleEditor:numberOfChildrenForCriterion:withRowType:` (page 3795) *required method*
  Returns the number of child items of a given criterion or row type. (required)
- `ruleEditor:predicatePartsForCriterion:withDisplayValue:inRow:` (page 3795)
  Returns a dictionary representing the parts of the predicate determined by the given criterion and value.

Monitoring Row Changes

- `ruleEditorRowsDidChange:` (page 3796)
  Notifies the receiver that a rule editor's rows changed.

Instance Methods

ruleEditor:child:forCriterion:withRowType:

Returns the child of a given item at a given index. (required)

```
- (id)ruleEditor:(NSRuleEditor *)editor child:(NSInteger)index
    forCriterion:(id)criterion withRowType:(NSRuleEditorRowType)rowType
```

Parameters

editor

The rule editor that sent the message.

index

The index of the requested child criterion. This value must be in the range from `0` up to (but not including) the number of children, as reported by the delegate in `ruleEditor:numberOfChildrenForCriterion:withRowType:` (page 3795).

criterion

The parent of the requested child, or `nil` if the rule editor is requesting a root criterion.

rowType

The type of the row.

Return Value

An object representing the requested child (or root) criterion. This object is used by the delegate to represent that position in the tree, and is passed as a parameter in subsequent calls to the delegate.

Special Considerations

The delegate must implement this method.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSRuleEditor.h`

ruleEditor:displayValueForCriterion:inRow:

Returns the value for a given criterion. (required)

```
- (id)ruleEditor:(NSRuleEditor *)editor displayValueForCriterion:(id)criterion
    inRow:(NSInteger)row
```

Parameters

editor

The rule editor that sent the message.

criterion

The criterion for which the value is required.

row

The row number of *criterion*.

Return Value
The value for *criterion*.

Discussion
The value should be an instance of `NSString`, `NSView`, or `NSMenuItem`. If the value is an `NSView` or `NSMenuItem`, you must ensure it is unique for every invocation of this method; that is, do not return a particular instance of `NSView` or `NSMenuItem` more than once.

Special Considerations

The delegate must implement this method.

Availability
Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSRuleEditor.h`

ruleEditor:numberOfChildrenForCriterion:withRowType:

Returns the number of child items of a given criterion or row type. (required)

```
- (NSInteger)ruleEditor:(NSRuleEditor *)editor
    numberOfChildrenForCriterion:(id)criterion
    withRowType:(NSRuleEditorRowType)rowType
```

Parameters

editor

The rule editor that sent the message.

criterion

The criterion for which the number of children is required.

rowType

The type of row of *criterion*.

Return Value
The number of child items of *criterion*. If *criterion* is `nil`, return the number of root criteria for the row type *rowType*.

Special Considerations

The delegate must implement this method.

Availability
Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSRuleEditor.h`

ruleEditor:predicatePartsForCriterion:withDisplayValue:inRow:

Returns a dictionary representing the parts of the predicate determined by the given criterion and value.

```
- (NSDictionary *)ruleEditor:(NSRuleEditor *)editor
    predicatePartsForCriterion:(id)criterion withDisplayValue:(id)value
    inRow:(NSInteger)row
```

Parameters

editor

The rule editor that sent the message.

criterion

The criterion for which the predicate parts are required.

value

The display value.

row

The row number of *criterion*.

Return Value

A dictionary representing the parts of the predicate determined by the given criterion and value. The keys of the dictionary should be the string constants specified in `Predicate Part Keys` (page 2230) with corresponding appropriate values.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSRuleEditor.h`

ruleEditorRowsDidChange:

Notifies the receiver that a rule editor's rows changed.

```
- (void)ruleEditorRowsDidChange:(NSNotification *)notification
```

Parameters

notification

A notification named`NSRuleEditorRowsDidChangeNotification` (page 2231).

Discussion

If the delegate implements this method, `NSRuleEditor` automatically registers its delegate to receive `NSRuleEditorRowsDidChangeNotification` (page 2231) notifications.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSRuleEditor.h`

NSServicesRequests Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSApplication.h
Companion guide	Services Implementation Guide

Overview

This informal protocol consists of two methods, `writeSelectionToPasteboard:types:` (page 3798) and `readSelectionFromPasteboard:` (page 3797). The first method provides data to a remote service, and the second receives any data the remote service might send back. Both respond to messages that are generated when the user chooses a command from the Services menu.

Tasks

Working with Pasteboards

- `readSelectionFromPasteboard:` (page 3797)
  Reads data from the pasteboard and uses it to replace the current selection.
- `writeSelectionToPasteboard:types:` (page 3798)
  Writes the current selection to the pasteboard.

Instance Methods

readSelectionFromPasteboard:

Reads data from the pasteboard and uses it to replace the current selection.

```
- (BOOL)readSelectionFromPasteboard:(NSPasteboard *)pboard
```

Parameters

pboard
  The pasteboard containing the data to read.

Return Value

`YES` if your implementation was able to read the pasteboard data successfully; otherwise, `NO`.

Discussion
You implement this method to replace your application's current selection (that is, the text or objects that are currently selected) with the data on the pasteboard. The data would have been placed in the pasteboard by another application in response to a remote message from the Services menu. A `readSelectionFromPasteboard:` message is sent to the same object that previously received a `writeSelectionToPasteboard:types:` (page 3798) message.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

writeSelectionToPasteboard:types:

Writes the current selection to the pasteboard.

```
- (BOOL)writeSelectionToPasteboard:(NSPasteboard *)pboard types:(NSArray *)types
```

Parameters

pboard
The pasteboard to receive your data.

types
An array of `NSString` objects listing the types of data that you should write to the pasteboard. You should write data to the pasteboard for as many of the types as you support.

Return Value
`YES` if your implementation was able to write one or more types to the pasteboard; otherwise, `NO`.

Discussion
A `writeSelectionToPasteboard:types:` message is sent to the first responder when the user chooses a command from the Services menu, but only if the receiver didn't return `nil` to a previous `validRequestorForSendType:returnType:` (page 2204) message.

After your method writes the data to the pasteboard, a remote message is sent to the application that provides the service the user requested. If the service provider supplies return data to replace the selection, the first responder will then receive a `readSelectionFromPasteboard:` (page 3797) message.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- validRequestorForSendType:returnType:` (page 2204) (NSResponder)

Related Sample Code
iSpend

Declared In
`NSApplication.h`

NSSoundDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSSound.h
Companion guide	Sound Programming Topics for Cocoa

Overview

The `NSSoundDelegate` protocol defines the optional methods implemented by delegates of `NSSound` objects.

Tasks

Playing Sounds

`- sound:didFinishPlaying:` (page 3799)
This delegate method is called when an `NSSound` instance has completed playback of its sound data.

Instance Methods

sound:didFinishPlaying:

This delegate method is called when an `NSSound` instance has completed playback of its sound data.

```
- (void)sound:(NSSound *)sound didFinishPlaying:(BOOL)finishedPlaying
```

Parameters

sound
The `NSSound` that has completed playback of its sound data.

finishedPlaying
`YES` when playback was successful; `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSSound.h`

NSSpeechRecognizerDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSSpeechRecognizer.h
Companion guide	Speech

Overview

The `NSSpeechRecognizerDelegate` protocol defines the optional methods implemented by delegates of `NSSpeechRecognizer` objects.

Tasks

Recognizing Commands

`- speechRecognizer:didRecognizeCommand:` (page 3801)
Invoked when the recognition engine has recognized the application command *command*.

Instance Methods

speechRecognizer:didRecognizeCommand:

Invoked when the recognition engine has recognized the application command *command*.

```
- (void)speechRecognizer:(NSSpeechRecognizer *)sender didRecognizeCommand:(id)command
```

Discussion
command is one of the strings from the array passed to `setCommands:` (page 2483). The delegate typically evaluates which command was recognized and performs the related action.

Availability
Available in Mac OS X v10.3 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSSpeechRecognizer.h`

NSSpeechSynthesizerDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSSpeechSynthesizer.h
Companion guide	Speech

Overview

The `NSSpeechSynthesizerDelegate` protocol defines the optional methods implemented by delegates of `NSSpeechSynthesizer` objects.

Tasks

Synthesizing Speech

- `speechSynthesizer:willSpeakWord:ofString:` (page 3806)
  Sent just before a synthesized word is spoken through the sound output device.
- `speechSynthesizer:willSpeakPhoneme:` (page 3805)
  Sent just before a synthesized phoneme is spoken through the sound output device.
- `speechSynthesizer:didEncounterErrorAtIndex:ofString:message:` (page 3804)
  Sent to the delegate when a speech synthesizer encounters an error in text being synthesized.
- `speechSynthesizer:didEncounterSyncMessage:` (page 3804)
  Sent to the delegate when a speech synthesizer encounters a synchronization error.
- `speechSynthesizer:didFinishSpeaking:` (page 3805)
  Sent when an `NSSpeechSynthesizer` object finishes speaking through the sound output device.

Instance Methods

speechSynthesizer:didEncounterErrorAtIndex:ofString:message:

Sent to the delegate when a speech synthesizer encounters an error in text being synthesized.

```
- (void)speechSynthesizer:(NSSpeechSynthesizer *)sender
    didEncounterErrorAtIndex:(NSUInteger)characterIndex ofString:(NSString *)string
     message:(NSString *)message
```

Parameters

sender

Speech synthesizer informing its delegate of an error.

characterIndex

Location in text where the receiver encountered the error.

text

Text the receiver was synthesizing when the error occurred.

errorMessage

Error message.

Discussion

The synthesizer sends an error delegate message whenever it encounters a syntax error within a command embedded in the string it is processing. This can be useful during application debugging, to detect problems with commands that you have embedded in strings that your application speaks. It can also be useful if your application allows users to embed commands within strings. Your application might display an alert indicating that the synthesizer encountered a problem in processing an embedded command.

If your application needs information about errors that occurred prior to calling your error delegate method, the application (including the error delegate method) can call the sender's `objectForProperty:error:` (page 2494) method with the `NSSpeechErrorsProperty` (page 2505) constant.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSSpeechSynthesizer.h`

speechSynthesizer:didEncounterSyncMessage:

Sent to the delegate when a speech synthesizer encounters a synchronization error.

```
- (void)speechSynthesizer:(NSSpeechSynthesizer *)sender
    didEncounterSyncMessage:(NSString *)message
```

Parameters

sender

Speech synthesizer informing its delegate of an error.

errorMessage

Error message.

Discussion
The synthesizer calls your synchronization delegate method whenever it encounters a synchronization command embedded in a string. You might use the synchronization delegate method to provide a callback not ordinarily provided.

For example, you might insert synchronization commands at the end of every sentence in a string, or you might enter synchronization commands after every numeric value in the text.

However, to synchronize your application with phonemes or words, it makes more sense to use the built-in phoneme and word delegate methods: `speechSynthesizer:willSpeakPhoneme:` (page 3805) and `speechSynthesizer:willSpeakWord:ofString:` (page 3806).

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSSpeechSynthesizer.h`

speechSynthesizer:didFinishSpeaking:

Sent when an `NSSpeechSynthesizer` object finishes speaking through the sound output device.

```
- (void)speechSynthesizer:(NSSpeechSynthesizer *)sender
    didFinishSpeaking:(BOOL)success
```

Parameters

sender
An `NSSpeechSynthesizer` object that has stopped speaking into the sound output device.

success
`YES` when speaking completed normally, `NO` if speaking is stopped prematurely for any reason.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `startSpeakingString:` (page 2498) (NSSpeechSynthesizer)
- `stopSpeaking` (page 2500) (NSSpeechSynthesizer)

Declared In
`NSSpeechSynthesizer.h`

speechSynthesizer:willSpeakPhoneme:

Sent just before a synthesized phoneme is spoken through the sound output device.

```
- (void)speechSynthesizer:(NSSpeechSynthesizer *)sender
    willSpeakPhoneme:(short)phonemeOpcode
```

Parameters

sender

An `NSSpeechSynthesizer` object that's synthesizing text into speech.

phonemeOpcode

Phoneme that *sender* is about to speak into the sound output device.

Discussion

One use of this method might be to animate a mouth on screen to match the generated speech.

Important: In Mac OS X v10.4 and earlier, the delegate is not sent this message when the `NSSpeechSynthesizer` object is synthesizing speech to a file (`startSpeakingString:toURL:` (page 2499)).

Availability

Available in Mac OS X v10.3 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

`- startSpeakingString:` (page 2498) (NSSpeechSynthesizer)

Declared In

`NSSpeechSynthesizer.h`

speechSynthesizer:willSpeakWord:ofString:

Sent just before a synthesized word is spoken through the sound output device.

```
- (void)speechSynthesizer:(NSSpeechSynthesizer *)sender
    willSpeakWord:(NSRange)wordToSpeak ofString:(NSString *)text
```

Parameters

sender

An `NSSpeechSynthesizer` object that's synthesizing text into speech.

wordToSpeak

Word that *sender* is about to speak into the sound output device.

text

Text that is being synthesized by *sender*.

Discussion

One use of this method might be to visually highlight the word being spoken.

Important: In Mac OS X v10.4 and earlier, the delegate is not sent this message when the `NSSpeechSynthesizer` object is synthesizing speech to a file (`startSpeakingString:toURL:` (page 2499)).

Availability

Available in Mac OS X v10.3 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- startSpeakingString:` (page 2498) (NSSpeechSynthesizer)

Declared In
`NSSpeechSynthesizer.h`

NSSplitViewDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSSplitView.h
Companion guides	Scroll View Programming Guide for Cocoa View Programming Guide
Related sample code	CoreAnimationText

Overview

The `NSSplitViewDelegate` protocol defines the optional methods implemented by delegates of `NSSplitView` objects.

Tasks

Managing Subviews

- `splitView:resizeSubviewsWithOldSize:` (page 3814)
  Allows the delegate to specify custom sizing behavior for the subviews of the NSSplitView *sender*.
- `splitViewWillResizeSubviews:` (page 3817)
  Invoked by the default notification center to notify the delegate that the splitview will resize its subviews.
- `splitViewDidResizeSubviews:` (page 3817)
  Invoked by the default notification center to notify the delegate that the splitview did resize its subviews.
- `splitView:canCollapseSubview:` (page 3811)
  Allows the delegate to determine whether the user can collapse and uncollapse *subview*.
- `splitView:shouldCollapseSubview:forDoubleClickOnDividerAtIndex:` (page 3815)
  Invoked to allow a delegate to determine if a subview should collapse in response to a double click.
- `splitView:shouldAdjustSizeOfSubview:` (page 3815)
  Allows the delegate to specify whether the subview should be resized.

Configuring and Drawing View Dividers

- `splitView:effectiveRect:forDrawnRect:ofDividerAtIndex:` (page 3813)
  Allows the delegate to modify the rectangle in which mouse clicks initiate divider dragging.
- `splitView:shouldHideDividerAtIndex:` (page 3816)
  Allows the delegate to determine whether a divider can be dragged or adjusted off the edge of the split view.
- `splitView:additionalEffectiveRectOfDividerAtIndex:` (page 3810)
  Allows the delegate to return an additional rectangle in which mouse clicks will initiate divider dragging.

Constraining Split Position

- `splitView:constrainMaxCoordinate:ofSubviewAt:` (page 3811)
  Allows the delegate for *sender* to constrain the maximum coordinate limit of a divider when the user drags it.
- `splitView:constrainMinCoordinate:ofSubviewAt:` (page 3812)
  Allows the delegate for *sender* to constrain the minimum coordinate limit of a divider when the user drags it.
- `splitView:constrainSplitPosition:ofSubviewAt:` (page 3813)
  Allows the delegate for *sender* to constrain the divider to certain positions.

Instance Methods

splitView:additionalEffectiveRectOfDividerAtIndex:

Allows the delegate to return an additional rectangle in which mouse clicks will initiate divider dragging.

```
- (NSRect)splitView:(NSSplitView *)splitView
    additionalEffectiveRectOfDividerAtIndex:(NSInteger)dividerIndex
```

Parameters

splitView
The split view that sent the message.

dividerIndex
The index of the divider.

Return Value
An additional rectangle in which mouse clicks should initiate divider dragging. The rectangle should be expressed in the coordinate system defined by *splitView*. Returning `NSZeroRect` indicates no additional dragging rectangle is desired.

Discussion
If a split view has no delegate, or if its delegate does not respond to this message, only mouse clicks within the effective frame of a divider initiate divider dragging.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSSplitView.h

splitView:canCollapseSubview:

Allows the delegate to determine whether the user can collapse and uncollapse *subview*.

```
- (BOOL)splitView:(NSSplitView *)splitView canCollapseSubview:(NSView *)subview
```

Parameters

splitView
The split view that sent the message.

subview
The subview to collapse.

Return Value
YES if *subview* should collapse when the user drags a divider beyond the halfway mark between its minimum size and its edge, otherwise NO.

Discussion
The *subview* uncollapses when the user drags the divider back beyond the halfway mark between its minimum size and its edge.

To specify the minimum size, define the methods splitView:constrainMaxCoordinate:ofSubviewAt: (page 3811) and splitView:constrainMinCoordinate:ofSubviewAt: (page 3812). A subview can collapse only if you also define splitView:constrainMinCoordinate:ofSubviewAt: (page 3812).

A collapsed subview is hidden but retained by the split view object, with the same size it had before it was collapsed.

If the delegate does not implement this method the subviews can't be collapsed.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSSplitView.h

splitView:constrainMaxCoordinate:ofSubviewAt:

Allows the delegate for *sender* to constrain the maximum coordinate limit of a divider when the user drags it.

```
- (CGFloat)splitView:(NSSplitView *)splitView
    constrainMaxCoordinate:(CGFloat)proposedMax ofSubviewAt:(NSInteger)dividerIndex
```

Parameters

splitView
The split view that sent the message.

proposedMax
The proposed maximum coordinate limit of the subview, in the split view's flipped coordinate system.

dividerIndex
Specifies the divider the user is moving, with the first divider being 0 and increasing from top to bottom (or left to right).

Return Value
The maximum coordinate limit of the divider.

Discussion
This method is invoked before the split view begins tracking the mouse to position a divider. You may further constrain the limits that have been already set, but you cannot extend the divider limits.

If the split bars are horizontal (views are one on top of the other), *proposedMax* is the bottom limit. If the split bars are vertical (views are side by side), *proposedMax* is the right limit. The initial value of *proposedMax* is the bottom (or right side) of the subview after the divider.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- isVertical (page 2545) (NSSplitView)

Declared In
NSSplitView.h

splitView:constrainMinCoordinate:ofSubviewAt:

Allows the delegate for *sender* to constrain the minimum coordinate limit of a divider when the user drags it.

```
- (CGFloat)splitView:(NSSplitView *)splitView
    constrainMinCoordinate:(CGFloat)proposedMin ofSubviewAt:(NSInteger)dividerIndex
```

Parameters

splitView
The split view that sent the message.

proposedMin
The proposed minimum coordinate limit of the subview, in the split view's flipped coordinate system.

dividerIndex
Specifies the divider the user is moving, with the first divider being 0 and increasing from top to bottom (or left to right).

Return Value
The minimum coordinate limit of the divider.

Discussion
This method is invoked before the split view begins tracking the cursor to position a divider. You may further constrain the limits that have been already set, but you cannot extend the divider limits.

If the split bars are horizontal (views are one on top of the other), *proposedMin* is the top limit. If the split bars are vertical (views are side by side), *proposedMin* is the left limit. The initial value of *proposedMin* is the top (or left side) of the subview before the divider.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `isVertical` (page 2545) (NSSplitView)

Declared In
`NSSplitView.h`

splitView:constrainSplitPosition:ofSubviewAt:

Allows the delegate for *sender* to constrain the divider to certain positions.

```
- (CGFloat)splitView:(NSSplitView *)splitView
    constrainSplitPosition:(CGFloat)proposedPosition
    ofSubviewAt:(NSInteger)dividerIndex
```

Parameters

splitView
The split view that sent the message.

proposedPosition
The cursor's current position, and the proposed position of the divider.

dividerIndex
Specifies the divider the user is moving, with the first divider being 0 and increasing from top to bottom (or left to right).

Return Value
The position at which to constrain the divider.

Discussion
If the delegate implements this method, the split view calls it repeatedly as the user moves the divider.

For example, if a subview's height must be a multiple of a certain number, use this method to return the multiple nearest to *proposedPosition*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSSplitView.h`

splitView:effectiveRect:forDrawnRect:ofDividerAtIndex:

Allows the delegate to modify the rectangle in which mouse clicks initiate divider dragging.

```
- (NSRect)splitView:(NSSplitView *)splitView
    effectiveRect:(NSRect)proposedEffectiveRect forDrawnRect:(NSRect)drawnRect
    ofDividerAtIndex:(NSInteger)dividerIndex
```

Parameters

splitView

The split view that sent the message.

proposedEffectiveRect

The proposed rectangle in which mouse clicks should initiate divider dragging. The rectangle is expressed in the coordinate system defined by *splitView*.

drawnRect

The frame of the divider, expressed in the coordinate system defined by *splitView*.

dividerIndex

The index of the divider.

Return Value

The rectangle in which mouse clicks should initiate divider dragging. The rectangle should be expressed in the coordinate system defined by *splitView*.

Discussion

A split view with thick dividers proposes the drawn frame as the effective frame. A split view with thin dividers proposes an effective frame that's a little larger than the drawn frame, to make it easier for the user to actually grab the divider.

If a split view has no delegate, or if its delegate does not respond to this message, the split view behaves as if it has a delegate that returns *proposedEffectiveRect* when sent this message.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSSplitView.h

splitView:resizeSubviewsWithOldSize:

Allows the delegate to specify custom sizing behavior for the subviews of the NSSplitView *sender*.

```
- (void)splitView:(NSSplitView *)splitView resizeSubviewsWithOldSize:(NSSize)oldSize
```

Parameters

splitView

The split view that sent the message.

oldSize

The size of the split view before the user resized it.

Discussion

If the delegate implements this method, `splitView:resizeSubviewsWithOldSize:` (page 3814) is invoked after the split view is resized.

The subviews should be resized such that the sum of the sizes of the subviews plus the sum of the thickness of the dividers equals the size of the NSSplitView's new frame. You can get the thickness of a divider through the `dividerThickness` (page 2543) method.

Note that if you implement this delegate method to resize subviews on your own, the NSSplitView does not perform any error checking for you. However, you can invoke `adjustSubviews` (page 2541) to perform the default sizing behavior.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `adjustSubviews` (page 2541) (NSSplitView)
- `setFrame:` (page 3218) (NSView)

Declared In
`NSSplitView.h`

splitView:shouldAdjustSizeOfSubview:

Allows the delegate to specify whether the subview should be resized.

```
- (BOOL)splitView:(NSSplitView *)splitView shouldAdjustSizeOfSubview:(NSView
    *)subview
```

Parameters

splitView
The split view that sent the message.

subview
The subview to resize.

Return Value
`YES` if adjustSubviews can change the size of the subview, otherwise `NO`. By returning `NO`, you lock the size of the split view *subview* while the split view is resized.

Discussion
Regardless of the value returned by this method, `adjustSubviews` (page 2541) may change the origin of the subview. Otherwise non-resizable subviews may also be resized in order to prevent an invalid subview layout.

If a split view has no delegate, or if its delegate does not respond to this message, the split view behaves as if this method returns `YES`.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSSplitView.h`

splitView:shouldCollapseSubview:forDoubleClickOnDividerAtIndex:

Invoked to allow a delegate to determine if a subview should collapse in response to a double click.

```
- (BOOL)splitView:(NSSplitView *)splitView shouldCollapseSubview:(NSView *)subview
    forDoubleClickOnDividerAtIndex:(NSInteger)dividerIndex
```

Parameters

splitView
The split view that sent the message.

subview
The subview to collapse.

dividerIndex
The index of the divider.

Return Value
YES if the subview should collapse, NO otherwise.

Discussion
If implemented, the delegate will receive this message once for the subview before a divider when the user double-clicks on that divider, and again for the subview after the divider, but only if the delegate returned YES when sent splitView:canCollapseSubview: (page 3811) for the subview in question. When the delegate indicates that both subviews should be collapsed NSSplitView's behavior is undefined.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSSplitView.h

splitView:shouldHideDividerAtIndex:

Allows the delegate to determine whether a divider can be dragged or adjusted off the edge of the split view.

```
- (BOOL)splitView:(NSSplitView *)splitView
    shouldHideDividerAtIndex:(NSInteger)dividerIndex
```

Parameters

splitView
The split view that sent the message.

dividerIndex
The zero-based index of the divider.

Return Value
YES if the divider should allow dragging off the edge of the split view, resulting in it not being visible.

Discussion
If a split view has no delegate, or if its delegate does not respond to this message, the split view behaves as if it has a delegate that returns NO when sent this message.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSSplitView.h

splitViewDidResizeSubviews:

Invoked by the default notification center to notify the delegate that the splitview did resize its subviews.

```
- (void)splitViewDidResizeSubviews:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSSplitViewDidResizeSubviewsNotification` (page 2549).

Discussion

If the delegate implements this method, the delegate is automatically registered to receive this notification.

This method is invoked after the split view resizes two of its subviews in response to the repositioning of a divider.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSSplitView.h`

splitViewWillResizeSubviews:

Invoked by the default notification center to notify the delegate that the splitview will resize its subviews.

```
- (void)splitViewWillResizeSubviews:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSSplitViewWillResizeSubviewsNotification` (page 2550).

Discussion

If the delegate implements this method, the delegate is automatically registered to receive this notification.

This method is invoked before the split view resizes two of its subviews in response to the repositioning of a divider.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSSplitView.h`

NSTableViewDataSource Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSTableView.h
Companion guide	Table View Programming Guide
Related sample code	AnimatedTableView ClockControl Cocoa Tips and Tricks QuickLookDownloader

Overview

The `NSTableViewDataSource` protocol declares the methods that an instance of `NSTableView` uses to provide and access the contents of its data source object.

Note: Some of the methods in this protocol, such as `tableView:objectValueForTableColumn:row:` (page 3822) and `numberOfRowsInTableView:` (page 3820) along with other methods that return data, are called very frequently, so they must be efficient.

Tasks

Getting Values

- `numberOfRowsInTableView:` (page 3820)
  Returns the number of records managed for *`aTableView`* by the data source object.
- `tableView:objectValueForTableColumn:row:` (page 3822)
  Invoked by the table view to return the data object associated with the specified row and column.

Setting Values

`- tableView:setObjectValue:forTableColumn:row:` (page 3823)
Sets the data object for an item in a given row in a given column.

Dragging

`- tableView:acceptDrop:row:dropOperation:` (page 3821)
Invoked by *aTableView* when the mouse button is released over a table view that previously decided to allow a drop.

`- tableView:namesOfPromisedFilesDroppedAtDestination:forDraggedRowsWithIndexes:` (page 3821)
Returns an array of filenames that represent the *indexSet* rows for a drag to *dropDestination*.

`- tableView:validateDrop:proposedRow:proposedDropOperation:` (page 3824)
Used by *aTableView* to determine a valid drop target.

`- tableView:writeRowsWithIndexes:toPasteboard:` (page 3825)
Returns a Boolean value that indicates whether a drag operation is allowed.

Sorting

`- tableView:sortDescriptorsDidChange:` (page 3823)
Invoked by *aTableView* to indicate that sorting may need to be done.

Instance Methods

numberOfRowsInTableView:

Returns the number of records managed for *aTableView* by the data source object.

```
- (NSInteger)numberOfRowsInTableView:(NSTableView *)aTableView
```

Parameters

aTableView
The table view that sent the message.

Return Value
The number of rows in *aTableView*.

Discussion
An instance of `NSTableView` uses this method to determine how many rows it should create and display. Your `numberOfRowsInTableView:` implementation is called very frequently, so it must be efficient.

Note: This method is optional if your application is using Cocoa bindings for providing data to the table view, otherwise it must be implemented.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTableView.h`

tableView:acceptDrop:row:dropOperation:

Invoked by *aTableView* when the mouse button is released over a table view that previously decided to allow a drop.

```
- (BOOL)tableView:(NSTableView *)aTableView acceptDrop:(id < NSDraggingInfo >)info
    row:(NSInteger)row dropOperation:(NSTableViewDropOperation)operation
```

Parameters

aTableView
The table view that sent the message.

info
An object that contains more information about this dragging operation.

row
The index of the proposed target row.

operation
The type of dragging operation.

Return Value
`YES` if the drop operation was successful, otherwise `NO`.

Discussion
The data source should incorporate the data from the dragging pasteboard in the implementation of this method. You can get the data for the drop operation from info using the `draggingPasteboard` (page 3662) method.

To accept a drop on the second row, *row* would be 2 and *operation* would be `NSTableViewDropOn`. To accept a drop below the last row, *row* would be `[aTableView numberOfRows]` and *operation* would be `NSTableViewDropAbove`.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTableView.h`

tableView:namesOfPromisedFilesDroppedAtDestination:forDraggedRowsWithIndexes:

Returns an array of filenames that represent the *indexSet* rows for a drag to *dropDestination*.

```
- (NSArray *)tableView:(NSTableView *)aTableView
    namesOfPromisedFilesDroppedAtDestination:(NSURL *)dropDestination
    forDraggedRowsWithIndexes:(NSIndexSet *)indexSet
```

Parameters

aTableView

The table view that sent the message.

dropDestination

The drop location where the files are created.

indexSet

The indexes of the items being dragged.

Return Value

An array of filenames (not full paths) for the created files that the receiver promises to create.

Discussion

This method is called when a destination has accepted a promise drag.

For more information on file promise dragging, see documentation on the `NSDraggingSource` protocol and `namesOfPromisedFilesDroppedAtDestination:` (page 3663).

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTableView.h`

tableView:objectValueForTableColumn:row:

Invoked by the table view to return the data object associated with the specified row and column.

```
- (id)tableView:(NSTableView *)aTableView objectValueForTableColumn:(NSTableColumn
    *)aTableColumn row:(NSInteger)rowIndex
```

Parameters

aTableView

The table view that sent the message.

aTableColumn

A column in in *aTableView*.

rowIndex

The row of the item in *aTableColumn*.

Return Value

An item in the data source in the specified tableColumn of the view.

Discussion

`tableView:objectValueForTableColumn:row:` is called each time the table cell needs to be redisplayed, so it must be efficient.

Note: This method is optional if your application is using Cocoa bindings for providing data to the table view, otherwise it must be implemented.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTableView.h

tableView:setObjectValue:forTableColumn:row:

Sets the data object for an item in a given row in a given column.

```
- (void)tableView:(NSTableView *)aTableView setObjectValue:(id)anObject
    forTableColumn:(NSTableColumn *)aTableColumn row:(NSInteger)rowIndex
```

Parameters

aTableView
The table view that sent the message.

anObject
The new value for the item.

aTableColumn
A column in *aTableView*.

rowIndex
The row of the item in *aTableColumn*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTableView.h

tableView:sortDescriptorsDidChange:

Invoked by *aTableView* to indicate that sorting may need to be done.

```
- (void)tableView:(NSTableView *)aTableView sortDescriptorsDidChange:(NSArray
    *)oldDescriptors
```

Parameters

aTableView
The table view that sent the message.

oldDescriptors
An array that contains the previous descriptors.

Discussion
The data source typically sorts and reloads the data, and adjusts the selections accordingly. If you need to know the current sort descriptors and the data source does not manage them itself, you can get the current sort descriptors by sending *aTableView* a `sortDescriptors` (page 2670) message.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTableView.h`

tableView:validateDrop:proposedRow:proposedDropOperation:

Used by *aTableView* to determine a valid drop target.

```
- (NSDragOperation)tableView:(NSTableView *)aTableView validateDrop:(id <
    NSDraggingInfo >)info proposedRow:(NSInteger)row
    proposedDropOperation:(NSTableViewDropOperation)operation
```

Parameters

aTableView
The table view that sent the message.

info
An object that contains more information about this dragging operation.

row
The index of the proposed target row.

operation
The type of dragging operation proposed.

Return Value
The dragging operation the data source will perform.

Discussion
The data source may "retarget" a drop if desired by calling `setDropRow:dropOperation:` (page 2663) and returning something other than `NSDragOperationNone`. One may choose to retarget for various reasons (e.g. for better visual feedback when inserting into a sorted position).

To propose a drop on the second row, *row* would be 2 and *operation* would be `NSTableViewDropOn`. To propose a drop below the last row, *row* would be `[aTableView numberOfRows]` and *operation* would be `NSTableViewDropAbove`.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTableView.h`

tableView:writeRowsWithIndexes:toPasteboard:

Returns a Boolean value that indicates whether a drag operation is allowed.

```
- (BOOL)tableView:(NSTableView *)aTableView writeRowsWithIndexes:(NSIndexSet
    *)rowIndexes toPasteboard:(NSPasteboard *)pboard
```

Parameters

aTableView

The table view that sent the message.

rowIndexes

An index set of row numbers that will be participating in the drag.

pboard

The pasteboard to which to write the drag data.

Return Value

YES if the drag operation is allowed, *NO* otherwise.

Discussion

Invoked by *aTableView* after it has been determined that a drag should begin, but before the drag has been started.

To refuse the drag, return `NO`. To start a drag, return `YES` and place the drag data onto *pboard* (data, owner, and so on). The drag image and other drag-related information will be set up and provided by the table view once this call returns with `YES`.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTableView.h`

NSTableViewDelegate Protocol Reference

Conforms to	NSControlTextEditingDelegate
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSTableView.h
Companion guides	Table View Programming Guide Drag and Drop Programming Topics for Cocoa
Related sample code	AnimatedTableView Cocoa Tips and Tricks DemoMonkey QuickLookDownloader

Overview

The `NSTableViewDelegate` protocol defines the optional methods implemented by delegates of `NSTableView` objects.

Tasks

Displaying Cells

- `tableView:willDisplayCell:forTableColumn:row:` (page 3840)
  Informs the delegate that the tableview will display the specified cell at the row in the column.
- `tableView:dataCellForTableColumn:row:` (page 3829)
  Invoked to allow the delegate to return a custom data cell for a specified row and column.
- `tableView:shouldShowCellExpansionForTableColumn:row:` (page 3836)
  Invoked to allow the delegate to control cell expansion for a specific row and column.
- `tableView:isGroupRow:` (page 3832)
  Invoked to allow the delegate to indicate that a specified row is a group row.
- `tableView:toolTipForCell:rect:tableColumn:row:mouseLocation:` (page 3838)
  Returns a string that is displayed as a tooltip for the specified cell in the column and row.

Editing Cells

- `tableView:shouldEditTableColumn:row:` (page 3834)
  Returns whether the cell at the specified row and column can be edited.

Setting Row and Column Size

- `tableView:heightOfRow:` (page 3831)
  Returns the height of the specified row.
- `tableView:sizeToFitWidthOfColumn:` (page 3838)
  Invoked to allow the delegate to provide custom sizing behavior when a column's resize divider is double clicked.

Selecting in the Tableview

- `selectionShouldChangeInTableView:` (page 3829)
  Returns whether the selection should change.
- `tableView:shouldSelectRow:` (page 3835)
  Returns whether the table view should allow selection of the specified row.
- `tableView:selectionIndexesForProposedSelection:` (page 3833)
  Invoked to allow the delegate to modify the proposed selection.
- `tableView:shouldSelectTableColumn:` (page 3835)
  Returns whether the specified table column can be selected.
- `tableViewSelectionIsChanging:` (page 3841)
  Informs the delegate that the table view's selection is in the process of changing (typically because the user is dragging the mouse across a number of rows).
- `tableViewSelectionDidChange:` (page 3841)
  Informs the delegate that the table view's selection has changed.
- `tableView:shouldTypeSelectForEvent:withCurrentSearchString:` (page 3837)
  Invoked to allow the delegate to control type select for a specific event.
- `tableView:typeSelectStringForTableColumn:row:` (page 3839)
  Invoked to allow the delegate to provide an alternate text value used for type selection for a specified row and column.
- `tableView:nextTypeSelectMatchFromRow:toRow:forString:` (page 3833)
  Invoked to allow the delegate to allow the delegate to modify how type selection works.

Moving and Resizing Columns

- `tableView:shouldReorderColumn:toColumn:` (page 3834)
  Sent to the delegate to allow or prohibit the specified column to be dragged to a new location.
- `tableView:didDragTableColumn:` (page 3831)
  Sent at the time the mouse button goes up in *`tableView`* and *`tableColumn`* has been dragged during the time the mouse button was down.

- `tableViewColumnDidMove:` (page 3840)
  Informs the delegate that a column was moved by user action in the table view.
- `tableViewColumnDidResize:` (page 3841)
  Informs the delegate that a column was resized in the table view.

Responding to Mouse Events

- `tableView:didClickTableColumn:` (page 3830)
  Sent when the mouse button was clicked in a table column, but the column was not dragged.
- `tableView:mouseDownInHeaderOfTableColumn:` (page 3832)
  Sent to the delegate whenever the mouse button is clicked in the table view's header column.
- `tableView:shouldTrackCell:forTableColumn:row:` (page 3837)
  Invoked to allow the delegate to control the tracking behavior for a specific cell.

Instance Methods

selectionShouldChangeInTableView:

Returns whether the selection should change.

```
- (BOOL)selectionShouldChangeInTableView:(NSTableView *)aTableView
```

Parameters

aTableView
  The table view that sent the message.

Return Value
`YES` to allow the table view to change its selection (typically a row being edited), `NO` to deny selection change.

Discussion
The user can select and edit different cells within the same row, but can't select another row unless the delegate approves. The delegate can implement this method for complex validation of edited rows based on the values of any of their cells.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTableView.h`

tableView:dataCellForTableColumn:row:

Invoked to allow the delegate to return a custom data cell for a specified row and column.

```
- (NSCell *)tableView:(NSTableView *)tableView dataCellForTableColumn:(NSTableColumn
    *)tableColumn row:(NSInteger)row
```

Parameters

tableView

The table view that sent the message.

tableColumn

The table column.

row

The row index.

Return Value

An `NSCell` subclass that is used for the specified *row* and *tableColumn*. The returned cell must properly implement `copyWithZone:`.

Discussion

A different data cell can be returned for any particular table column and row, or a cell that will be used for the entire row (a full width cell).

If the tableColumn is non-`nil`, you should return a cell, and generally that will be the result of sending *tableColumn* a `dataCellForRow:` (page 2590) message.

This method will be invoked with a *tableColumn* value of `nil` to allow you to return a group cell–a cell that will be used to draw the entire row, acting as a separator. If you return a cell when *tableColumn* is `nil`, any implemented datasource and delegate methods must be prepared to handle a `nil` table column value.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTableView.h`

tableView:didClickTableColumn:

Sent when the mouse button was clicked in a table column, but the column was not dragged.

```
- (void)tableView:(NSTableView *)tableView didClickTableColumn:(NSTableColumn
    *)tableColumn
```

Parameters

tableView

The table view that sent the message.

tableColumn

The table column.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTableView.h`

tableView:didDragTableColumn:

Sent at the time the mouse button goes up in *tableView* and *tableColumn* has been dragged during the time the mouse button was down.

```
- (void)tableView:(NSTableView *)tableView didDragTableColumn:(NSTableColumn
    *)tableColumn
```

Parameters

tableView

The table view that sent the message.

tableColumn

The table column.

Special Considerations

The behavior of this method on Mac OS X v10.5 is different from prior versions. On Mac OS X v 10.5 the dragged column is sent to the delegate. In earlier versions the table column that is currently located at the dragged column's original index is sent.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSTableView.h

tableView:heightOfRow:

Returns the height of the specified row.

```
- (CGFloat)tableView:(NSTableView *)tableView heightOfRow:(NSInteger)row
```

Parameters

tableView

The table view that sent the message.

row

The row index.

Return Value

The height of the row. The height should not include intercell spacing and must be greater than zero.

Discussion

You should implement this method if your table supports varying row heights.

Although table views may cache the returned values, you should ensure that this method is efficient. When you change a row's height you must invalidate the existing row height by calling `noteHeightOfRowsWithIndexesChanged:` (page ?). `NSTableView` automatically invalidates its entire row height cache when `reloadData` (page ?) and `noteNumberOfRowsChanged` (page ?) are called.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTableView.h

tableView:isGroupRow:

Invoked to allow the delegate to indicate that a specified row is a group row.

```
- (BOOL)tableView:(NSTableView *)tableView isGroupRow:(NSInteger)row
```

Parameters

tableView
The table view that sent the message.

row
The row index.

Return Value
YES if the specified row should have the group row style drawn, NO otherwise.

Discussion
If the cell in *row* is an NSTextFieldCell and contains only a string, the group row style attributes will automatically be applied to the cell.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTableView.h

tableView:mouseDownInHeaderOfTableColumn:

Sent to the delegate whenever the mouse button is clicked in the table view's header column.

```
- (void)tableView:(NSTableView *)tableView
    mouseDownInHeaderOfTableColumn:(NSTableColumn *)tableColumn
```

Parameters

tableView
The table view that sent the message.

tableColumn
The table column.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTableView.h

tableView:nextTypeSelectMatchFromRow:toRow:forString:

Invoked to allow the delegate to allow the delegate to modify how type selection works.

```
- (NSInteger)tableView:(NSTableView *)tableView
    nextTypeSelectMatchFromRow:(NSInteger)startRow toRow:(NSInteger)endRow
    forString:(NSString *)searchString
```

Parameters

tableView

The table view that sent the message.

startRow

The starting row of the search range.

endRow

The ending row of the search range.

searchString

A string containing the typed selection.

Return Value

The first row in the range of *startRow* through *endRow* (excluding *endRow* itself) that matches *selectionString*. Return -1 if no match is found.

Discussion

It is possible for `endRow` to be less than `startRow` if the search will wrap.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTableView.h`

tableView:selectionIndexesForProposedSelection:

Invoked to allow the delegate to modify the proposed selection.

```
- (NSIndexSet *)tableView:(NSTableView *)tableView
    selectionIndexesForProposedSelection:(NSIndexSet *)proposedSelectionIndexes
```

Parameters

tableView

The table view that sent the message.

proposedSelectionIndexes

An index set containing the indexes of the proposed selection.

Return Value

An `NSIndexSet` instance containing the indexes of the new selection. Return *proposedSelectionIndexes* if the proposed selection is acceptable, or the value of the table view's existing selection to avoid changing the selection.

Discussion

This method may be called multiple times with one new index added to the existing selection to find out if a particular index can be selected when the user is extending the selection with the keyboard or mouse.

Implementation of this method is optional. If implemented, this method will be called instead of `tableView:shouldSelectRow:` (page ?).

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTableView.h`

tableView:shouldEditTableColumn:row:

Returns whether the cell at the specified row and column can be edited.

```
- (BOOL)tableView:(NSTableView *)aTableView shouldEditTableColumn:(NSTableColumn
    *)aTableColumn row:(NSInteger)rowIndex
```

Parameters

aTableView
The table view that sent the message.

aTableColumn
The table column.

rowIndex
The row index.

Return Value
`YES` to allow editing the cell , `NO` to deny editing.

Discussion
The delegate can implement this method to disallow editing of specific cells.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTableView.h`

tableView:shouldReorderColumn:toColumn:

Sent to the delegate to allow or prohibit the specified column to be dragged to a new location.

```
- (BOOL)tableView:(NSTableView *)tableView shouldReorderColumn:(NSInteger)columnIndex
    toColumn:(NSInteger)newColumnIndex
```

Parameters

tableView
The tableview that sent the message.

columnIndex
The index of the column being dragged.

newColumnIndex
The proposed target index of the column.

Return Value
YES if the column reordering should be allowed, otherwise NO.

Discussion
When a column is initially dragged by the user, the delegate is first called with a *newColumnIndex* value of -1. Returning NO will disallow that column from being reordered at all. Returning YES allows it to be reordered, and the delegate will be called again when the column reaches a new location.

The actual NSTableColumn instance can be retrieved from the tableColumns (page 2670) array.

If this method is not implemented, all columns are considered reorderable.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSTableView.h

tableView:shouldSelectRow:

Returns whether the table view should allow selection of the specified row.

```
- (BOOL)tableView:(NSTableView *)aTableView shouldSelectRow:(NSInteger)rowIndex
```

Parameters
aTableView
The table view that sent the message.

rowIndex
The row index.

Return Value
YES to permit selection of the row, NO to deny selection.

Discussion
The delegate can implement this method to disallow selection of particular rows.

For better performance and finer-grain control over the selection, use tableView:selectionIndexesForProposedSelection: (page 2643).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTableView.h

tableView:shouldSelectTableColumn:

Returns whether the specified table column can be selected.

```
- (BOOL)tableView:(NSTableView *)aTableView shouldSelectTableColumn:(NSTableColumn
    *)aTableColumn
```

Parameters

aTableView

The table view that sent the message.

aTableColumn

The table column.

Return Value

`YES` to permit selection, otherwise `NO`.

Discussion

The delegate can implement this method to disallow selection of particular columns.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTableView.h`

tableView:shouldShowCellExpansionForTableColumn:row:

Invoked to allow the delegate to control cell expansion for a specific row and column.

```
- (BOOL)tableView:(NSTableView *)tableView
    shouldShowCellExpansionForTableColumn:(NSTableColumn *)tableColumn
    row:(NSInteger)row
```

Parameters

tableView

The table view that sent the message.

tableColumn

The table column.

row

The row index.

Return Value

`YES` if the tooltip cell should expand, `NO` otherwise.

Discussion

Cell expansion can occur when the mouse hovers over the specified cell and the cell contents are unable to be fully displayed within the cell. If this method returns `YES`, the full cell contents will be shown in a special floating tool tip view, otherwise the content is truncated.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTableView.h`

tableView:shouldTrackCell:forTableColumn:row:

Invoked to allow the delegate to control the tracking behavior for a specific cell.

```
- (BOOL)tableView:(NSTableView *)tableView shouldTrackCell:(NSCell *)cell
    forTableColumn:(NSTableColumn *)tableColumn row:(NSInteger)row
```

Parameters

tableView

The table view that sent the message.

cell

The cell to track.

tableColumn

The table column.

row

A row in *tableView*.

Return Value

`YES` if the cell should track, `NO` otherwise.

Discussion

Normally, only selectable or selected cells can be tracked. If you implement this method, cells which are not selectable or selected can be tracked, and vice-versa.

For example, this allows you to have an `NSButtonCell` in a table which does not change the selection, but can still be clicked on and tracked.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTableView.h`

tableView:shouldTypeSelectForEvent:withCurrentSearchString:

Invoked to allow the delegate to control type select for a specific event.

```
- (BOOL)tableView:(NSTableView *)tableView shouldTypeSelectForEvent:(NSEvent *)event
    withCurrentSearchString:(NSString *)searchString
```

Parameters

tableView

The table view that sent the message.

event

The event.

searchString

The search string or `nil` if no type select has began.

Return Value

YES to allow type select for event, NO otherwise.

Discussion
Typically, this is called from the table view `keyDown:` implementation and the event will be a key event.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTableView.h`

tableView:sizeToFitWidthOfColumn:

Invoked to allow the delegate to provide custom sizing behavior when a column's resize divider is double clicked.

```
- (CGFloat)tableView:(NSTableView *)tableView
    sizeToFitWidthOfColumn:(NSInteger)column
```

Parameters
tableView
The tableview that sent the message.

column
The index of the column.

Return Value
The width of the specified column.

Discussion
By default, `NSOutlineView` iterates every row in the table, accesses a cell via `preparedCellAtColumn:row:` (page 2643), and requests the `cellSize` (page 549) to find the appropriate largest width to use.

For accurate results and performance, it is recommended that this method is implemented when using large tables. By default, large tables use a monte carlo simulation instead of interating every row.

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTableView.h`

tableView:toolTipForCell:rect:tableColumn:row:mouseLocation:

Returns a string that is displayed as a tooltip for the specified cell in the column and row.

```
- (NSString *)tableView:(NSTableView *)aTableView toolTipForCell:(NSCell *)aCell
    rect:(NSRectPointer)rect tableColumn:(NSTableColumn *)aTableColumn
    row:(NSInteger)row mouseLocation:(NSPoint)mouseLocation
```

Parameters
aTableView
The table view that sent the message.

aCell
The cell.

rect
The proposed active area of the tooltip. You can modify *rect* to provide an alternative active area.

aTableColumn
The table column.

row
The row index.

mouseLocation
The mouse location.

Return Value
A string containing the tooltip. Return `nil` or the empty string if no tooltip is desired.

Discussion
By default, *rect* is computed as `[cell drawingRectForBounds:cellFrame]`.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTableView.h`

tableView:typeSelectStringForTableColumn:row:

Invoked to allow the delegate to provide an alternate text value used for type selection for a specified row and column.

```
- (NSString *)tableView:(NSTableView *)tableView
    typeSelectStringForTableColumn:(NSTableColumn *)tableColumn row:(NSInteger)row
```

Parameters
tableView
The table view that sent the message.

tableColumn
The table column.

row
The row index.

Return Value
A string that is used in type select comparison for *row* and *tableColumn*. Return `nil` if the *row* or *tableColumn* should not be searched.

Discussion
Implement this method to change the string value that is searched for based on what is displayed. By default, all cells with text in them are searched.

If this delegate method is not implemented the string value is:

```
[[tableView preparedCellAtColumn:tableColumn row:row] stringValue]
```

This value can be returned from the delegate method if desired.

Implementation of this method is optional.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTableView.h

tableView:willDisplayCell:forTableColumn:row:

Informs the delegate that the tableview will display the specified cell at the row in the column.

```
- (void)tableView:(NSTableView *)aTableView willDisplayCell:(id)aCell
    forTableColumn:(NSTableColumn *)aTableColumn row:(NSInteger)rowIndex
```

Parameters

aTableView
The table view that sent the message.

aCell
The cell to be displayed.

aTableColumn
The table column.

rowIndex
The row index.

Discussion
The delegate can modify the display attributes of *aCell* to alter the appearance of the cell.

Because *aCell* is reused for every row in *aTableColumn*, the delegate must set the display attributes both when drawing special cells and when drawing normal cells.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTableView.h

tableViewColumnDidMove:

Informs the delegate that a column was moved by user action in the table view.

```
- (void)tableViewColumnDidMove:(NSNotification *)aNotification
```

Parameters

aNotification
A notification named NSTableViewColumnDidMoveNotification (page ?).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTableView.h

tableViewColumnDidResize:

Informs the delegate that a column was resized in the table view.

```
- (void)tableViewColumnDidResize:(NSNotification *)aNotification
```

Parameters
aNotification
A notification named NSTableViewColumnDidResizeNotification (page ?).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTableView.h

tableViewSelectionDidChange:

Informs the delegate that the table view's selection has changed.

```
- (void)tableViewSelectionDidChange:(NSNotification *)aNotification
```

Parameters
aNotification
A notification named NSTableViewSelectionDidChangeNotification (page ?).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTableView.h

tableViewSelectionIsChanging:

Informs the delegate that the table view's selection is in the process of changing (typically because the user is dragging the mouse across a number of rows).

```
- (void)tableViewSelectionIsChanging:(NSNotification *)aNotification
```

Parameters
aNotification
A notification named NSTableViewSelectionIsChangingNotification (page ?).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTableView.h`

NSTabViewDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSTabView.h
Companion guide	Tab Views

Overview

The `NSTabViewDelegate` protocol defines the optional methods implemented by delegates of `NSTabView` objects.

Tasks

Adding and Removing Tabs

- `tabViewDidChangeNumberOfTabViewItems:` (page 3845)
  Informs the delegate that the number of tab view items in *`tabView`* has changed.

Selecting a Tab

- `tabView:shouldSelectTabViewItem:` (page 3844)
  Invoked just before *`tabViewItem`* in *`tabView`* is selected.
- `tabView:willSelectTabViewItem:` (page 3844)
  Informs the delegate that *`tabView`* is about to select *`tabViewItem`*.
- `tabView:didSelectTabViewItem:` (page 3844)
  Informs the delegate that *`tabView`* has selected *`tabViewItem`*.

Instance Methods

tabView:didSelectTabViewItem:

Informs the delegate that *tabView* has selected *tabViewItem*.

```
- (void)tabView:(NSTabView *)tabView didSelectTabViewItem:(NSTabViewItem
    *)tabViewItem
```

Parameters

tabView

The tab view that sent the request.

tabViewItem

The tab view item that was selected.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSTabView.h

tabView:shouldSelectTabViewItem:

Invoked just before *tabViewItem* in *tabView* is selected.

```
- (BOOL)tabView:(NSTabView *)tabView shouldSelectTabViewItem:(NSTabViewItem
    *)tabViewItem
```

Parameters

tabView

The tab view that sent the request.

tabViewItem

The tab view item to select.

Return Value

YES if the tab view item should be selected, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSTabView.h

tabView:willSelectTabViewItem:

Informs the delegate that *tabView* is about to select *tabViewItem*.

```
- (void)tabView:(NSTabView *)tabView willSelectTabViewItem:(NSTabViewItem
    *)tabViewItem
```

Parameters

tabView

The tab view that sent the request.

tabViewItem

The tab view item that is about to be selected.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSTabView.h

tabViewDidChangeNumberOfTabViewItems:

Informs the delegate that the number of tab view items in *tabView* has changed.

```
- (void)tabViewDidChangeNumberOfTabViewItems:(NSTabView *)tabView
```

Parameters

tabView

The tab view that added or removed tabview items.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- numberOfTabViewItems (page 2690)

Declared In

NSTabView.h

NSTextAttachmentCell Protocol Reference

Adopted by	NSTextAttachmentCell
Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSTextAttachment.h
Companion guides	Text System Overview Text Attachment Programming Topics

Overview

The NSTextAttachmentCell protocol declares the interface for objects that draw text attachment icons and handle mouse events on their icons. With the exceptions of `cellBaselineOffset` (page 3849), `setAttachment:` (page 3851), and `attachment` (page 3848), all of these methods are implemented by the NSCell class and described in that class specification.

See the NSAttributedString and NSTextView class specifications for general information on text attachments.

Tasks

Drawing

- `drawWithFrame:inView:` (page 3850) *required method*
  Draws the receiver's image within *cellFrame* in *aView*, which is the view currently focused. (required)
- `drawWithFrame:inView:characterIndex:` (page 3850) *required method*
  Draws the receiver's image within *cellFrame* in *aView*, which is the view currently focused. *charIndex* is the index of the attachment character within the text. (required)
- `drawWithFrame:inView:characterIndex:layoutManager:` (page 3850) *required method*
  Draws the receiver's image within *cellFrame* in *controlView*, which is the view currently focused. *charIndex* is the index of the attachment character within the text. *layoutManager* is the layout manager for the text. (required)

- highlight:withFrame:inView: (page 3851) *required method*
  Draws the receiver's image—with highlighting if *flag* is YES—within *cellFrame* in *aView*, which should be the focus view. (required)

Cell Size and Position

- cellSize (page 3849) *required method*
  Returns the size of the attachment's icon. (required)
- cellBaselineOffset (page 3849) *required method*
  Returns the position where the attachment cell's image should be drawn in text, relative to the current point established in the glyph layout. (required)
- cellFrameForTextContainer:proposedLineFragment:glyphPosition:characterIndex: (page 3849) *required method*
  Returns the frame of the cell as it would be drawn as the character at the given glyph *position*, and character index, *charIndex*, in *textContainer*. (required)

Event Handling

- wantsToTrackMouse (page 3853) *required method*
  Returns YES if the receiver will handle a mouse event occurring over its image (to support dragging, for example), NO otherwise. (required)
- wantsToTrackMouseForEvent:inRect:ofView:atCharacterIndex: (page 3853) *required method*
  Allows an attachment to specify what events it would want to track the mouse for. (required)
- trackMouse:inRect:ofView:untilMouseUp: (page 3852) *required method*
  Handles a mouse-down event on the receiver's image. (required)
- trackMouse:inRect:ofView:atCharacterIndex:untilMouseUp: (page 3851) *required method*
  Handles a mouse-down event on the receiver's image. (required)

Setting the Attachment

- setAttachment: (page 3851) *required method*
  Sets the text attachment object that owns the receiver to *anAttachment*, without retaining it (the text attachment, as the owner, retains the cell). (required)
- attachment (page 3848) *required method*
  Returns the text attachment object that owns the receiver. (required)

Instance Methods

attachment

Returns the text attachment object that owns the receiver. (required)

```
- (NSTextAttachment *)attachment
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAttachment:` (page 3851)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSTextAttachment.h`

cellBaselineOffset

Returns the position where the attachment cell's image should be drawn in text, relative to the current point established in the glyph layout. (required)

```
- (NSPoint)cellBaselineOffset
```

Discussion
The image should be drawn so its lower-left corner lies on this point.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `icon` (NSFileWrapper)

Declared In
`NSTextAttachment.h`

cellFrameForTextContainer:proposedLineFragment:glyphPosition:characterIndex:

Returns the frame of the cell as it would be drawn as the character at the given glyph *position*, and character index, *charIndex*, in *textContainer*. (required)

```
- (NSRect)cellFrameForTextContainer:(NSTextContainer *)textContainer
    proposedLineFragment:(NSRect)lineFrag glyphPosition:(NSPoint)position
    characterIndex:(NSUInteger)charIndex
```

Discussion
The proposed line fragment is specified by *lineFrag*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextAttachment.h`

cellSize

Returns the size of the attachment's icon. (required)

```
- (NSSize)cellSize
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `icon` (NSFileWrapper)
- `fileWrapper` (page 2757) (NSTextAttachment)

Declared In
`NSTextAttachment.h`

drawWithFrame:inView:

Draws the receiver's image within *cellFrame* in *aView*, which is the view currently focused. (required)

```
- (void)drawWithFrame:(NSRect)cellFrame inView:(NSView *)aView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `drawWithFrame:inView:` (page 554) (NSCell)
- `lockFocus` (page 3187) (NSView)

Declared In
`NSTextAttachment.h`

drawWithFrame:inView:characterIndex:

Draws the receiver's image within *cellFrame* in *aView*, which is the view currently focused. *charIndex* is the index of the attachment character within the text. (required)

```
- (void)drawWithFrame:(NSRect)cellFrame inView:(NSView *)aView
    characterIndex:(NSUInteger)charIndex
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextAttachment.h`

drawWithFrame:inView:characterIndex:layoutManager:

Draws the receiver's image within *cellFrame* in *controlView*, which is the view currently focused. *charIndex* is the index of the attachment character within the text. *layoutManager* is the layout manager for the text. (required)

```
- (void)drawWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
    characterIndex:(NSUInteger)charIndex layoutManager:(NSLayoutManager
    *)layoutManager
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextAttachment.h

highlight:withFrame:inView:

Draws the receiver's image—with highlighting if *flag* is YES—within *cellFrame* in *aView*, which should be the focus view. (required)

```
- (void)highlight:(BOOL)flag withFrame:(NSRect)cellFrame inView:(NSView *)aView
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- highlight:withFrame:inView: (page 560) (NSCell)
- lockFocus (page 3187) (NSView)

Declared In
NSTextAttachment.h

setAttachment:

Sets the text attachment object that owns the receiver to *anAttachment*, without retaining it (the text attachment, as the owner, retains the cell). (required)

```
- (void)setAttachment:(NSTextAttachment *)anAttachment
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- attachment (page 3848)
- setAttachmentCell: (page 2758) (NSTextAttachment)

Declared In
NSTextAttachment.h

trackMouse:inRect:ofView:atCharacterIndex:untilMouseUp:

Handles a mouse-down event on the receiver's image. (required)

```
- (BOOL)trackMouse:(NSEvent *)theEvent inRect:(NSRect)cellFrame ofView:(NSView
    *)aTextView atCharacterIndex:(NSUInteger)charIndex untilMouseUp:(BOOL)flag
```

Discussion
theEvent is the mouse-down event. *cellFrame* is the region of *aTextView* in which you should track further mouse events. *charIndex* is the position in the text at which this attachment appears. *aTextView* is the view that received the event. It's assumed to be an NSTextView, and should be the focus view. If *flag*

is YES, the receiver tracks the mouse until a mouse-up event occurs; if *flag* is NO, it stops tracking when a mouse-dragged event occurs outside of *cellFrame*. Returns YES if the receiver successfully finished tracking the mouse (typically through a mouse-up event), NO otherwise (such as when the mouse is dragged outside *cellFrame*).

NSTextAttachmentCell's implementation of this method calls upon *aTextView*'s delegate to handle the event. If *theEvent* is a mouse-up event for a double click, the text attachment cell sends the delegate a textView:doubleClickedOnCell:inRect: message and returns YES. Otherwise, depending on whether the user clicks or drags the cell, it sends the delegate a textView:clickedOnCell:inRect: or a textView:draggedCell:inRect:event: message and returns YES. NSTextAttachmentCell's implementation returns NO only if *flag* is NO and the mouse is dragged outside of *cellFrame*. The delegate methods are invoked only if the delegate responds.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSTextAttachment.h

trackMouse:inRect:ofView:untilMouseUp:

Handles a mouse-down event on the receiver's image. (required)

```
- (BOOL)trackMouse:(NSEvent *)theEvent inRect:(NSRect)cellFrame ofView:(NSView
    *)aTextView untilMouseUp:(BOOL)flag
```

Discussion
theEvent is the mouse-down event. *cellFrame* is the region of *aTextView* in which further mouse events should be tracked. *aTextView* is the view that received the event. It's assumed to be an NSTextView and should be the focus view. If *flag* is YES, the receiver tracks the mouse until a mouse-up event occurs; if *flag* is NO, it stops tracking when a mouse-dragged event occurs outside of *cellFrame*. Returns YES if the receiver successfully finished tracking the mouse (typically through a mouse-up event), NO otherwise (such as when the cursor is dragged outside *cellFrame*).

NSTextAttachmentCell's implementation of this method calls upon the delegate of *aTextView* to handle the event. If *theEvent* is a mouse-up event for a double click, the text attachment cell sends the delegate a textView:doubleClickedOnCell:inRect: message and returns YES. Otherwise, depending on whether the user clicks or drags the cell, it sends the delegate a textView:clickedOnCell:inRect: or a textView:draggedCell:inRect:event: message and returns YES. NSTextAttachmentCell's implementation returns NO only if *flag* is NO and the cursor is dragged outside of *cellFrame*. The delegate methods are invoked only if the delegate responds.

Availability
Available in Mac OS X v10.0 and later.

See Also
- wantsToTrackMouse (page 3853)
- trackMouse:inRect:ofView:untilMouseUp: (page 610) (NSCell)
- lockFocus (page 3187) (NSView)

Declared In
NSTextAttachment.h

wantsToTrackMouse

Returns `YES` if the receiver will handle a mouse event occurring over its image (to support dragging, for example), `NO` otherwise. (required)

```
- (BOOL)wantsToTrackMouse
```

Discussion
NSTextAttachmentCell's implementation of this method returns `YES`. The NSView containing the cell should invoke this method before sending a `trackMouse:inRect:ofView:untilMouseUp:` (page 3852) message.

For an attachment in an attributed string, if the attachment cell returns `NO` its attachment character should be selected rather than the cell being asked to track the mouse. This results in the attachment icon behaving as any regular glyph in text.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextAttachment.h`

wantsToTrackMouseForEvent:inRect:ofView:atCharacterIndex:

Allows an attachment to specify what events it would want to track the mouse for. (required)

```
- (BOOL)wantsToTrackMouseForEvent:(NSEvent *)theEvent inRect:(NSRect)cellFrame
    ofView:(NSView *)controlView atCharacterIndex:(NSUInteger)charIndex
```

Discussion
`theEvent` is the event in question that occurred in *`cellFrame`* inside *`controlView`*. *`charIndex`* is the index of the attachment character within the text. If `wantsToTrackMouse` (page 3853) returns `YES`, this method allows the attachment to decide whether it wishes to do so for particular events.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSTextAttachment.h`

NSTextDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSText.h
Companion guide	Text System Overview

Overview

The `NSTextDelegate` protocol defines the optional methods implemented by delegates of `NSText` objects.

Tasks

Changing Text Formatting

- `textDidChange:` (page 3856)
  Informs the delegate that the text object has changed its characters or formatting attributes.

Editing Text

- `textShouldBeginEditing:` (page 3857)
  Invoked when a text object begins to change its text, this method requests permission for *`aTextObject`* to begin editing.
- `textDidBeginEditing:` (page 3856)
  Informs the delegate that the text object has begun editing (that the user has begun changing it).
- `textShouldEndEditing:` (page 3857)
  Invoked from a text object's implementation of `resignFirstResponder` (page 2189), this method requests permission for *`aTextObject`* to end editing.
- `textDidEndEditing:` (page 3856)
  Informs the delegate that the text object has finished editing (that it has resigned first responder status).

Instance Methods

textDidBeginEditing:

Informs the delegate that the text object has begun editing (that the user has begun changing it).

```
- (void)textDidBeginEditing:(NSNotification *)aNotification
```

Discussion
The name of *aNotification* is NSTextDidBeginEditingNotification (page 2752).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSText.h

textDidChange:

Informs the delegate that the text object has changed its characters or formatting attributes.

```
- (void)textDidChange:(NSNotification *)aNotification
```

Discussion
The name of *aNotification* is NSTextDidChangeNotification (page 2752).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSText.h

textDidEndEditing:

Informs the delegate that the text object has finished editing (that it has resigned first responder status).

```
- (void)textDidEndEditing:(NSNotification *)aNotification
```

Discussion
The name of *aNotification* is NSTextDidEndEditingNotification (page 2752).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSText.h

textShouldBeginEditing:

Invoked when a text object begins to change its text, this method requests permission for *aTextObject* to begin editing.

```
- (BOOL)textShouldBeginEditing:(NSText *)aTextObject
```

Discussion
If the delegate returns YES, the text object proceeds to make changes. If the delegate returns NO, the text object abandons the editing operation. This method is also invoked when the user drags and drops a file onto the text object.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- makeFirstResponder: (page 3344) (NSWindow)
- becomeFirstResponder (page 2144) (NSResponder)

Declared In
NSText.h

textShouldEndEditing:

Invoked from a text object's implementation of resignFirstResponder (page 2189), this method requests permission for *aTextObject* to end editing.

```
- (BOOL)textShouldEndEditing:(NSText *)aTextObject
```

Discussion
If the delegate returns YES, the text object proceeds to finish editing and resign first responder status. If the delegate returns NO, the text object selects all of its text and remains the first responder.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- resignFirstResponder (page 2189) (NSResponder)

Declared In
NSText.h

NSTextFieldDelegate Protocol Reference

Conforms to	NSControlTextEditingDelegate
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSTextField.h
Companion guide	Text Fields

Overview

The `NSTextFieldDelegate` protocol adopts the `NSControlTextEditingDelegate` protocol and currently does not extend it further.

NSTextInput Protocol Reference

Adopted by	NSInputManager
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSInputManager.h
Companion guides	Text System Overview Text Input Management

Overview

The `NSTextInput` protocol defines the methods that Cocoa text views must implement in order to interact properly with the text input management system.

Important: `NSTextInput` protocol is slated for deprecation. Please use `NSTextInputClient` protocol, introduced in Mac OS X v10.5, as described in *NSTextInputClient Protocol Reference*.

`NSTextView` and its abstract superclass `NSText` are the only classes included in Cocoa that implement `NSTextInput`. To create another text view class, you can either subclass `NSTextView` (and not `NSText`, for historical reasons), or subclass `NSView` and implement the `NSTextInput` protocol.

Important: Methods specific to the `NSTextInput` protocol are intended for dealing with text input and generally are not suitable for other purposes.

Tasks

Marked Text

- `hasMarkedText` (page 3865) *required method*
  Returns a Boolean value indicating whether or not the receiver has marked text. (required)
- `markedRange` (page 3865) *required method*
  Returns the range of the marked text. (required)

- `selectedRange` (page 3866) *required method*
  Returns the range of selected text. (required)
- `setMarkedText:selectedRange:` (page 3866) *required method*
  Replaces currently marked text in the receiver's text storage with the given string and sets the selection to the given range, computed from the beginning of the marked text. (required)
- `unmarkText` (page 3867) *required method*
  Removes any marking from pending input text and disposes of the marked text as it wishes. The text view should accept the marked text as if it had been inserted normally. (required)
- `validAttributesForMarkedText` (page 3867) *required method*
  Returns an array of names for the attributes supported by the receiver. (required)

Text Storage

- `attributedSubstringFromRange:` (page 3862) *required method*
  Returns an attributed string derived from the given range in the receiver's text storage. (required)
- `insertText:` (page 3865) *required method*
  Inserts the given string into the receiver's text storage. (required)

Character Coordinates

- `characterIndexForPoint:` (page 3863) *required method*
  Returns the index of the character whose frame rectangle includes the given point. (required)
- `firstRectForCharacterRange:` (page 3864) *required method*
  Returns the first frame rectangle for characters in the given range, in screen coordinates. (required)

Key Bindings

- `doCommandBySelector:` (page 3864) *required method*
  Invokes the given selector if possible. (required)

Other

- `conversationIdentifier` (page 3863) *required method*
  Returns a number used to identify the receiver's context to the input server. (required)

Instance Methods

attributedSubstringFromRange:

Returns an attributed string derived from the given range in the receiver's text storage. (required)

```
- (NSAttributedString *)attributedSubstringFromRange:(NSRange)theRange
```

Parameters

theRange

The range in the text storage from which to create the returned string.

Return Value

The string created from the given range.

Discussion

This method allows input mangers to query any range in text storage.

An implementation of this method should be prepared for *theRange* to be out-of-bounds. For example, the InkWell text input service can ask for the contents of the text input client that extends beyond the document's range. In this case, you should return the intersection of the document's range and *theRange*. If the location of *theRange* is completely outside of the document's range, return `nil`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSInputManager.h`

characterIndexForPoint:

Returns the index of the character whose frame rectangle includes the given point. (required)

```
- (NSUInteger)characterIndexForPoint:(NSPoint)thePoint
```

Parameters

thePoint

A point, in screen coordinates.

Return Value

The character index, measured from the start of the receiver's text storage, of the character containing the given point. Returns `NSNotFound` if the cursor is not within a character.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSInputManager.h`

conversationIdentifier

Returns a number used to identify the receiver's context to the input server. (required)

```
- (NSInteger)conversationIdentifier
```

Return Value

The identifying number of the receiver.

Discussion

Each text view within an application should return a unique identifier (typically its address). However, multiple text views sharing the same text storage must all return the same identifier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSInputManager.h`

doCommandBySelector:

Invokes the given selector if possible. (required)

```
- (void)doCommandBySelector:(SEL)aSelector
```

Parameters

aSelector
The selector to be invoked.

Discussion
If *aSelector* cannot be invoked, then `doCommandBySelector:` should not pass this message up the responder chain. `NSResponder` also implements this method, and it does forward uninvokable commands up the responder chain, but a text view should not. A text view implementing the NSTextInput protocol inherits from `NSView`, which inherits from `NSResponder`, so your implementation of this method will override the one in `NSResponder`. It should not call `super`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `interpretKeyEvents:` (page 2158) (NSResponder)
- `doCommandBySelector:` (page 2152) (NSResponder)

Declared In
`NSInputManager.h`

firstRectForCharacterRange:

Returns the first frame rectangle for characters in the given range, in screen coordinates. (required)

```
- (NSRect)firstRectForCharacterRange:(NSRange)theRange
```

Parameters

theRange
The character range whose frame is returned.

Return Value
The frame rectangle for the given range of characters.

Discussion
If *theRange* spans multiple lines of text in the text view, the rectangle returned is the one for the characters in the first line. If the length of *theRange* is 0 (as it would be if there is nothing selected at the insertion point), the rectangle coincides with the insertion point, and its width is 0.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSInputManager.h

hasMarkedText

Returns a Boolean value indicating whether or not the receiver has marked text. (required)

```
- (BOOL)hasMarkedText
```

Return Value
YES if the receiver has marked text, NO if it doesn't.

Discussion
Unlike other methods in this protocol, this one is not called by an input server. The text view itself may call this method to determine whether there currently is marked text. NSTextView, for example, disables the Edit > Copy menu item when this method returns YES.

Availability
Available in Mac OS X v10.0 and later.

See Also
- markedRange (page 3865)

Declared In
NSInputManager.h

insertText:

Inserts the given string into the receiver's text storage. (required)

```
- (void)insertText:(id)aString
```

Parameters
aString
Either an NSString or an NSAttributedString object.

Discussion
This method is the entry point for inserting text typed by the user and is generally not suitable for other purposes. Programmatic modification of the text is best done by operating on the text storage directly. Because this method pertains to the actions of the user, the text view must be editable for the insertion to work.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSInputManager.h

markedRange

Returns the range of the marked text. (required)

```
- (NSRange)markedRange
```

Return Value
The range of marked text.

Discussion
The returned range measures from the start of the receiver's text storage. The return value's `location` is `NSNotFound`, and its `length` is 0 if and only if `hasMarkedText` (page 3865) returns `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMarkedText:selectedRange:` (page 3866)
- `unmarkText` (page 3867)
- `hasMarkedText` (page 3865)

Declared In
`NSInputManager.h`

selectedRange

Returns the range of selected text. (required)

```
- (NSRange)selectedRange
```

Return Value
The range of selected text.

Discussion
The returned range measures from the start of the receiver's text storage. If there is no selection, the return value's `location` is `NSNotFound`, and its `length` is 0.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setMarkedText:selectedRange:` (page 3866)

Declared In
`NSInputManager.h`

setMarkedText:selectedRange:

Replaces currently marked text in the receiver's text storage with the given string and sets the selection to the given range, computed from the beginning of the marked text. (required)

```
- (void)setMarkedText:(id)aString selectedRange:(NSRange)selRange
```

Parameters
aString
Either an `NSString` or an `NSAttributedString` object; must not be `nil`.

selRange
The range within *aString* to set as the selection.

Discussion
If there is no marked text, the current selection is replaced. If there is no selection, the string is inserted at the insertion point.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedRange` (page 3866)
- `unmarkText` (page 3867)

Declared In
`NSInputManager.h`

unmarkText

Removes any marking from pending input text and disposes of the marked text as it wishes. The text view should accept the marked text as if it had been inserted normally. (required)

```
- (void)unmarkText
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `selectedRange` (page 3866)
- `setMarkedText:selectedRange:` (page 3866)

Declared In
`NSInputManager.h`

validAttributesForMarkedText

Returns an array of names for the attributes supported by the receiver. (required)

```
- (NSArray *)validAttributesForMarkedText
```

Return Value
An array of `NSString` objects representing names for the supported attributes.

Discussion
The input server may choose to use some of these attributes in the text it inserts or in marked text. Returns an empty array if no attributes are supported. See NSAttributedString Additions (page 249) for the set of string constants that you could return in the array.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSInputManager.h`

NSTextInputClient Protocol Reference

Adopted by	NSTextView
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	AppKit/NSTextInputClient.h
Related sample code	TextInputView

Overview

The `NSTextInputClient` protocol defines the methods that Cocoa text views must implement in order to interact properly with the text input management system. To create another text view class, you can either subclass `NSTextView` (and not `NSText`, for historical reasons), or subclass `NSView` and implement the `NSTextInputClient` protocol

Important: Methods specific to the `NSTextInputClient` protocol are intended for dealing with text input and generally are not suitable for other purposes.

Tasks

Handling Marked Text

- `hasMarkedText` (page 3875) *required method*
  Returns a Boolean value indicating whether the receiver has marked text. (required)
- `markedRange` (page 3875) *required method*
  Returns the range of the marked text. (required)
- `selectedRange` (page 3876) *required method*
  Returns the range of selected text. (required)
- `setMarkedText:selectedRange:replacementRange:` (page 3876) *required method*
  Replaces a specified range in the receiver's text storage with the given string and sets the selection. (required)
- `unmarkText` (page 3877) *required method*
  Unmarks the marked text. (required)

- `validAttributesForMarkedText` (page 3878) *required method* Available in Mac OS X v10.0 through Mac OS X v10.1
  Returns an array of attribute names recognized by the receiver. (required)

Storing Text

- `attributedSubstringForProposedRange:actualRange:` (page 3871) *required method*
  Returns an attributed string derived from the given range in the receiver's text storage. (required)
- `insertText:replacementRange:` (page 3875) *required method*
  Inserts the given string into the receiver, replacing the specified content. (required)

Getting Character Coordinates

- `characterIndexForPoint:` (page 3872) *required method*
  Returns the index of the character whose bounding rectangle includes the given point. (required)
- `firstRectForCharacterRange:actualRange:` (page 3873) *required method*
  Returns the first logical boundary rectangle for characters in the given range. (required)

Binding Keystrokes

- `doCommandBySelector:` (page 3872) *required method*
  Invokes the action specified by the given selector. (required)

Optional Methods

- `attributedString` (page 3871)
  Returns an attributed string representing the receiver's text storage.
- `fractionOfDistanceThroughGlyphForPoint:` (page 3874)
  Returns the fraction of the distance from the left side of the character to the right side that a given point lies.
- `baselineDeltaForCharacterAtIndex:` (page 3872)
  Returns the baseline position of a given character relative to the origin of rectangle returned by `firstRectForCharacterRange:actualRange:` (page 3873).
- `windowLevel` (page 3878)
  Returns the window level of the receiver.
- `drawsVerticallyForCharacterAtIndex:` (page 3873) *required method*
  Informs the text input management system whether the protocol-conforming client renders the character at the given index vertically. (required)

Instance Methods

attributedString

Returns an attributed string representing the receiver's text storage.

```
- (NSAttributedString *)attributedString
```

Return Value
The attributed string of the receiver's text storage.

Discussion
Implementation of this method is optional. A class adopting the `NSTextInputClient` protocol can implement this interface if it can be done efficiently to enable callers of this interface to access arbitrary portions of the receiver's content more efficiently.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
TextInputView

Declared In
`NSTextInputClient.h`

attributedSubstringForProposedRange:actualRange:

Returns an attributed string derived from the given range in the receiver's text storage. (required)

```
- (NSAttributedString *)attributedSubstringForProposedRange:(NSRange)aRange
    actualRange:(NSRangePointer)actualRange
```

Parameters

aRange
The range in the text storage from which to create the returned string.

actualRange
The actual range of the returned string if it was adjusted, for example, to a grapheme cluster boundary or for performance or other reasons. `NULL` if range was not adjusted.

Return Value
The string created from the given range. May return `nil`.

Discussion
An implementation of this method should be prepared for *aRange* to be out of bounds. For example, the InkWell text input service can ask for the contents of the text input client that extends beyond the document's range. In this case, you should return the intersection of the document's range and *aRange*. If the location of *aRange* is completely outside of the document's range, return `nil`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTextInputClient.h

baselineDeltaForCharacterAtIndex:

Returns the baseline position of a given character relative to the origin of rectangle returned by `firstRectForCharacterRange:actualRange:` (page 3873).

```
- (CGFloat)baselineDeltaForCharacterAtIndex:(NSUInteger)anIndex
```

Parameters

anIndex

Index of the character whose baseline is tested.

Return Value
The vertical distance, in points, between the baseline of the character at *anIndex* and the rectangle origin.

Discussion
Implementation of this method is optional. This information allows the caller to determine finer-grained character positioning within the text storage of the text view adopting `NSTextInputClient`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTextInputClient.h

characterIndexForPoint:

Returns the index of the character whose bounding rectangle includes the given point. (required)

```
- (NSUInteger)characterIndexForPoint:(NSPoint)aPoint
```

Parameters

aPoint

The point to test, in screen coordinates.

Return Value
The character index, measured from the start of the receiver's text storage, of the character containing the given point. Returns `NSNotFound` if the cursor is not within a character's bounding rectangle.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTextInputClient.h

doCommandBySelector:

Invokes the action specified by the given selector. (required)

```
- (void)doCommandBySelector:(SEL)aSelector
```

Parameters

aSelector

The selector to invoke.

Discussion

If *aSelector* cannot be invoked, then `doCommandBySelector:` should not pass this message up the responder chain. `NSResponder` also implements this method, and it does forward uninvokable commands up the responder chain, but a text view should not. A text view implementing the `NSTextInputClient` protocol inherits from `NSView`, which inherits from `NSResponder`, so your implementation of this method will override the one in `NSResponder`. It should not call `super`.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `interpretKeyEvents:` (page 2158) (NSResponder)
- `doCommandBySelector:` (page 2152) (NSResponder)

Declared In

`NSTextInputClient.h`

drawsVerticallyForCharacterAtIndex:

Informs the text input management system whether the protocol-conforming client renders the character at the given index vertically. (required)

```
- (BOOL)drawsVerticallyForCharacterAtIndex:(NSUInteger)charIndex
```

Parameters

charIndex

The index of the character to test.

Return Value

`YES` if the character is rendered vertically; otherwise `NO`.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSTextInputClient.h`

firstRectForCharacterRange:actualRange:

Returns the first logical boundary rectangle for characters in the given range. (required)

```
- (NSRect)firstRectForCharacterRange:(NSRange)aRange
    actualRange:(NSRangePointer)actualRange
```

Parameters

aRange

The character range whose boundary rectangle is returned.

actualRange

If non-NULL, contains the character range corresponding to the returned area if it was adjusted, for example, to a grapheme cluster boundary or characters in the first line fragment.

Return Value
The boundary rectangle for the given range of characters, in screen coordinates. The rectangle's size value can be negative if the text flows to the left.

Discussion
If *aRange* spans multiple lines of text in the text view, the rectangle returned is the one surrounding the characters in the first line. In that case *actualRange* contains the range covered by the first rect, so you can query all line fragments by invoking this method repeatedly. If the length of *aRange* is 0 (as it would be if there is nothing selected at the insertion point), the rectangle coincides with the insertion point, and its width is 0.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTextInputClient.h

fractionOfDistanceThroughGlyphForPoint:

Returns the fraction of the distance from the left side of the character to the right side that a given point lies.

```
- (CGFloat)fractionOfDistanceThroughGlyphForPoint:(NSPoint)aPoint
```

Parameters
aPoint

The point to test.

Return Value
The fraction of the distance *aPoint* is through the glyph in which it lies. May be 0 or 1 if *aPoint* is not within the bounding rectangle of a glyph (0 if the point is to the left or above the glyph; 1 if it's to the right or below).

Discussion
Implementation of this method is optional. This allows caller to perform precise selection handling.

For purposes such as dragging out a selection or placing the insertion point, a partial percentage less than or equal to 0.5 indicates that *aPoint* should be considered as falling before the glyph; a partial percentage greater than 0.5 indicates that it should be considered as falling after the glyph. If the nearest glyph doesn't lie under *aPoint* at all (for example, if *aPoint* is beyond the beginning or end of a line), this ratio is 0 or 1.

For example, if the glyph stream contains the glyphs "A" and "b", with the width of "A" being 13 points, and *aPoint* is 8 points from the left side of "A", then the fraction of the distance is 8/13, or 0.615. In this case, the *aPoint* should be considered as falling between "A" and "b" for purposes such as dragging out a selection or placing the insertion point.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTextInputClient.h

hasMarkedText

Returns a Boolean value indicating whether the receiver has marked text. (required)

```
- (BOOL)hasMarkedText
```

Return Value
`YES` if the receiver has marked text; otherwise `NO`.

Discussion
The text view itself may call this method to determine whether there currently is marked text. `NSTextView`, for example, disables the Edit > Copy menu item when this method returns `YES`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `markedRange` (page 3875)

Related Sample Code
TextInputView

Declared In
`NSTextInputClient.h`

insertText:replacementRange:

Inserts the given string into the receiver, replacing the specified content. (required)

```
- (void)insertText:(id)aString replacementRange:(NSRange)replacementRange
```

Parameters
`aString`
The text to insert, either an `NSString` or `NSAttributedString` instance.

`replacementRange`
The range of content to replace in the receiver's text storage.

Discussion
This method is the entry point for inserting text typed by the user and is generally not suitable for other purposes. Programmatic modification of the text is best done by operating on the text storage directly. Because this method pertains to the actions of the user, the text view must be editable for the insertion to work.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSTextInputClient.h`

markedRange

Returns the range of the marked text. (required)

```
- (NSRange)markedRange
```

Return Value
The range of marked text or `{NSNotFound, 0}` if there is no marked range.

Discussion
The returned range measures from the start of the receiver's text storage. The return value's `location` is `NSNotFound` and its `length` is 0 if and only if `hasMarkedText` (page 3875) returns `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `hasMarkedText` (page 3875)
- `setMarkedText:selectedRange:replacementRange:` (page 3876)
- `unmarkText` (page 3877)

Related Sample Code
TextInputView

Declared In
`NSTextInputClient.h`

selectedRange

Returns the range of selected text. (required)

```
- (NSRange)selectedRange
```

Return Value
The range of selected text or `{NSNotFound, 0}` if there is no selection.

Discussion
The returned range measures from the start of the receiver's text storage, that is, from 0 to the document length.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setMarkedText:selectedRange:replacementRange:` (page 3876)

Related Sample Code
TextInputView

Declared In
`NSTextInputClient.h`

setMarkedText:selectedRange:replacementRange:

Replaces a specified range in the receiver's text storage with the given string and sets the selection. (required)

```
- (void)setMarkedText:(id)aString selectedRange:(NSRange)selectedRange
    replacementRange:(NSRange)replacementRange
```

Parameters

aString

The string to insert. Can be either an `NSString` or `NSAttributedString` instance.

selectedRange

The range to set as the selection, computed from the beginning of the inserted string.

replacementRange

The range to replace, computed from the beginning of the marked text.

Discussion

If there is no marked text, the current selection is replaced. If there is no selection, the string is inserted at the insertion point.

When *aString* is an `NSString` object, the receiver is expected to render the marked text with distinguishing appearance (for example, `NSTextView` renders with `markedTextAttributes` (page 2906)).

Availability

Available in Mac OS X v10.5 and later.

See Also

- `selectedRange` (page 3876)
- `unmarkText` (page 3877)

Declared In

`NSTextInputClient.h`

unmarkText

Unmarks the marked text. (required)

```
- (void)unmarkText
```

Discussion

The receiver removes any marking from pending input text and disposes of the marked text as it wishes. The text view should accept the marked text as if it had been inserted normally. If there is no marked text, the invocation of this method has no effect.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `selectedRange` (page 3876)
- `setMarkedText:selectedRange:replacementRange:` (page 3876)

Related Sample Code

TextInputView

Declared In

`NSTextInputClient.h`

validAttributesForMarkedText

Returns an array of attribute names recognized by the receiver. (required)

```
- (NSArray *)validAttributesForMarkedText
```

Return Value
An array of `NSString` objects representing names for the supported attributes.

Discussion
Returns an empty array if no attributes are supported. See *NSAttributedString Application Kit Additions Reference* for the set of string constants representing standard attributes.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
TextInputView

Declared In
`NSTextInputClient.h`

windowLevel

Returns the window level of the receiver.

```
- (NSInteger)windowLevel
```

Return Value
The window level of the receiver.

Discussion
Implementation of this method is optional. A class adopting `NSTextInputClient` can implement this interface to specify its window level if it is higher than `NSFloatingWindowLevel`.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
TextInputView

Declared In
`NSTextInputClient.h`

NSTextViewDelegate Protocol Reference

Conforms to	NSTextDelegate
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSTextView.h
Companion guides	Text System Overview Text System User Interface Layer Programming Guide
Related sample code	FunHouse TextSizingExample

Overview

The `NSTextViewDelegate` protocol defines the optional methods implemented by delegates of `NSTextView` objects.

Tasks

Accessing Text System Objects

`- undoManagerForTextView:` (page 3896)
Returns the undo manager for the specified text view.

Controlling Display

`- textView:willDisplayToolTip:forCharacterAtIndex:` (page 3893)
Returns the actual tooltip to display.

Managing the Selection

`- textView:willChangeSelectionFromCharacterRange:toCharacterRange:` (page 3891)
Returns the actual range to select.

- `textView:willChangeSelectionFromCharacterRanges:toCharacterRanges:` (page 3892)
  Returns the actual character ranges to select.
- `textViewDidChangeSelection:` (page 3895)
  Sent when the selection changes in the text view.

Managing the Pasteboard

- `textView:writablePasteboardTypesForCell:atIndex:` (page 3894)
  Returns the writable pasteboard types for a given cell.
- `textView:writeCell:atIndex:toPasteboard:type:` (page 3895)
  Returns whether data of the specified type for the given cell could be written to the specified pasteboard.

Setting Text Attributes

- `textView:shouldChangeTextInRange:replacementString:` (page 3889)
  Sent when a text view needs to determine if text in a specified range should be changed.
- `textView:shouldChangeTextInRanges:replacementStrings:` (page 3889)
  Sent when a text view needs to determine if text in an array of specified ranges should be changed.
- `textView:shouldChangeTypingAttributes:toAttributes:` (page 3890)
  Sent when the typing attributes are changed.
- `textViewDidChangeTypingAttributes:` (page 3896)
  Sent when a text view's typing attributes change.

Clicking and Pasting

- `textView:clickedOnCell:inRect:atIndex:` (page 3882)
  Sent when the user clicks a cell.
- `textView:doubleClickedOnCell:inRect:atIndex:` (page 3886)
  Sent when the user double-clicks a cell.
- `textView:clickedOnLink:atIndex:` (page 3883)
  Sent after the user clicks a link.

Working With the Spelling Checker

- `textView:shouldSetSpellingState:range:` (page 3890)
  Sent when the spelling state is changed.
- `textView:willCheckTextInRange:options:types:` (page 3893)
  Invoked to allow the delegate to modify the text checking process before it occurs.
- `textView:didCheckTextInRange:types:options:results:orthography:wordCount:` (page 3884)
  Invoked to allow the delegate to modify the text checking results after checking has occurred.

Dragging

- `textView:draggedCell:inRect:event:atIndex:` (page 3888)
  Sent when the user attempts to drag a cell.

Text Completion

- `textView:completions:forPartialWordRange:indexOfSelectedItem:` (page 3884)
  Returns the actual completions for a partial word.

Performing Commands

- `textView:doCommandBySelector:` (page 3885)
  Sent to allow the delegate to perform the command for the text view.

Contextual Menu Management

- `textView:menu:forEvent:atIndex:` (page 3888)
  Allows delegate to control the context menu returned by the text view.

Deprecated Methods

- `textView:clickedOnLink:` (page 3883)
  Sent after the user clicks on a link. (Deprecated. Use `textView:clickedOnLink:atIndex:` instead.)
- `textView:draggedCell:inRect:event:` (page 3887)
  Sent when the user attempts to drag a cell. (Deprecated. Use `textView:draggedCell:inRect:event:atIndex:` instead.)
- `textView:clickedOnCell:inRect:` (page 3881)
  Sent when the user clicks a cell. (Deprecated. Use `textView:clickedOnCell:inRect:atIndex:` instead.)
- `textView:doubleClickedOnCell:inRect:` (page 3886)
  Sent when the user double-clicks a cell. (Deprecated. Use `textView:doubleClickedOnCell:inRect:atIndex:` instead.)

Instance Methods

textView:clickedOnCell:inRect:

Sent when the user clicks a cell. (Deprecated. Use `textView:clickedOnCell:inRect:atIndex:` instead.)

```
- (void)textView:(NSTextView *)aTextView clickedOnCell:(id < NSTextAttachmentCell
    >)attachmentCell inRect:(NSRect)cellFrame
```

Parameters

aTextView
The text view sending the message.

attachmentCell
The cell clicked by the user.

cellFrame
The frame of the clicked cell.

Discussion
This message is only sent if `textView:clickedOnCell:inRect:atIndex:` is not implemented. Implement this method in order to track the mouse after a mouse click on a cell.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTextView.h`

textView:clickedOnCell:inRect:atIndex:

Sent when the user clicks a cell.

```
- (void)textView:(NSTextView *)aTextView clickedOnCell:(id < NSTextAttachmentCell
    >)cell inRect:(NSRect)cellFrame atIndex:(NSUInteger)charIndex
```

Parameters

aTextView
The text view sending the message.

cell
The cell clicked by the user.

cellFrame
The frame of the clicked cell.

charIndex
The character index of the clicked cell.

Discussion
The delegate can use this message as its cue to perform an action or select the attachment cell's character. *aTextView* is the first text view in a series shared by a layout manager, not necessarily the one that draws *cell*.

The delegate may subsequently receive a `textView:doubleClickedOnCell:inRect:atIndex:` (page 3886) message if the user continues to perform a double click.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `textView:doubleClickedOnCell:inRect:atIndex:` (page 3886)

Declared In
NSTextView.h

textView:clickedOnLink:

Sent after the user clicks on a link. (Deprecated. Use `textView:clickedOnLink:atIndex:` instead.)

```
- (BOOL)textView:(NSTextView *)aTextView clickedOnLink:(id)link
```

Parameters

aTextView
The text view sending the message.

link
The link that was clicked.

Discussion
This message is only sent if `textView:clickedOnLink:atIndex:` is not implemented.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `clickedOnLink:atIndex:` (page 2887) (NSTextView)
- `textView:clickedOnLink:atIndex:` (page 3883)

Declared In
NSTextView.h

textView:clickedOnLink:atIndex:

Sent after the user clicks a link.

```
- (BOOL)textView:(NSTextView *)aTextView clickedOnLink:(id)link
    atIndex:(NSUInteger)charIndex
```

Parameters

aTextView
The text view sending the message.

link
The link that was clicked; the value of `NSLinkAttributeName` (page 272).

charIndex
The character index where the click occurred, indexed within the text storage.

Return Value
`YES` if the click was handled; otherwise, `NO` to allow the next responder to handle it.

Discussion
The delegate can use this method to handle the click on the link. It is invoked by `clickedOnLink:atIndex:` (page 2887).

The *charIndex* parameter is a character index somewhere in the range of the link attribute. If the user actually physically clicked the link, then it should be the character that was originally clicked. In some cases a link may be opened indirectly or programmatically, in which case a character index somewhere in the range of the link attribute is supplied.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- clickedOnLink:atIndex: (page 2887) (NSTextView)

Declared In
NSTextView.h

textView:completions:forPartialWordRange:indexOfSelectedItem:

Returns the actual completions for a partial word.

```
- (NSArray *)textView:(NSTextView *)textView completions:(NSArray *)words
    forPartialWordRange:(NSRange)charRange indexOfSelectedItem:(NSInteger *)index
```

Parameters

textView
The text view sending the message.

words
The proposed array of completions.

charRange
The range of characters to be completed.

index
On return, the index of the initially selected completion. The default is 0, and –1 indicates no selection.

Return Value
The actual array of completions that will be presented for the partial word at the given range. Returning nil or a zero-length array suppresses completion.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTextView.h

textView:didCheckTextInRange:types:options:results:orthography:wordCount:

Invoked to allow the delegate to modify the text checking results after checking has occurred.

```
- (NSArray *)textView:(NSTextView *)view didCheckTextInRange:(NSRange)range
    types:(NSTextCheckingTypes)checkingTypes options:(NSDictionary *)options
    results:(NSArray *)results orthography:(NSOrthography *)orthography
    wordCount:(NSInteger)wordCount
```

Parameters

view
The text view sending the message.

range
The range that was checked.

checkingTypes
The type of checking that was performed. The possible constants are listed in `NSTextCheckingTypes` and can be combined using the C bit-wise `OR` operator to perform multiple checks at the same time.

options
A dictionary of values used during the checking process to perform. See `Spell Checking Option Dictionary Keys` (page 2537) for the supported values.

results
An array of `NSTextCheckingResult` instances.

orthography
The orthography of the text.

wordCount
The number of words checked.

Return Value
An array of `NSTextCheckingResult` instances. You can return the results array as is, or an altered array of `NSTextCheckingResult` objects.

Discussion
Invoked by `handleTextCheckingResults:forRange:types:options:orthography:wordCount:` (page 2894), this method allows observation of text checking, or modification of the results

Availability
Available in Mac OS X v10.6 and later.

Declared In
`NSTextView.h`

textView:doCommandBySelector:

Sent to allow the delegate to perform the command for the text view.

```
- (BOOL)textView:(NSTextView *)aTextView doCommandBySelector:(SEL)aSelector
```

Parameters

aTextView
The text view sending the message. This is the first text view in a series shared by a layout manager.

aSelector
The selector.

Return Value
`YES` indicates that the delegate handled the command and the text view will not attempt to perform it; `NO` indicates that the delegate did not handle the command the text view will attempt to perform it.

Discussion
This method is invoked by `NSTextView`'s `doCommandBySelector:` (page 2152) method.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTextView.h

textView:doubleClickedOnCell:inRect:

Sent when the user double-clicks a cell. (Deprecated. Use `textView:doubleClickedOnCell:inRect:atIndex:` instead.)

```
- (void)textView:(NSTextView *)aTextView doubleClickedOnCell:(id <
    NSTextAttachmentCell >)attachmentCell inRect:(NSRect)cellFrame
```

Parameters

aTextView
The text view sending the message.

cell
The cell double-clicked by the user.

cellFrame
The frame of the double-clicked cell.

Discussion
This message is only sent if `textView:doubleClickedOnCell:inRect:atIndex:` (page 3886) is not implemented. Implement this method in order to track the mouse after a mouse double-click on a cell.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `textView:doubleClickedOnCell:inRect:atIndex:` (page 3886)

Declared In
NSTextView.h

textView:doubleClickedOnCell:inRect:atIndex:

Sent when the user double-clicks a cell.

```
- (void)textView:(NSTextView *)aTextView doubleClickedOnCell:(id <
    NSTextAttachmentCell >)cell inRect:(NSRect)cellFrame
    atIndex:(NSUInteger)charIndex
```

Parameters

aTextView
The text view sending the message.

cell
The cell double-clicked by the user.

cellFrame
The frame of the double-clicked cell.

charIndex
The character index of the double-clicked cell.

Discussion
The delegate can use this message as its cue to perform an action, such as opening the file represented by the attachment. *aTextView* is the first text view in a series shared by a layout manager, not necessarily the one that draws *cell*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTextView.h

textView:draggedCell:inRect:event:

Sent when the user attempts to drag a cell. (Deprecated. Use `textView:draggedCell:inRect:event:atIndex:` instead.)

```
- (void)textView:(NSTextView *)aTextView draggedCell:(id < NSTextAttachmentCell
    >)cell inRect:(NSRect)aRect event:(NSEvent *)theEvent
```

Parameters
aTextView
The text view sending the message.

cell
The cell being dragged.

aRect
The rectangle from which the cell was dragged.

theEvent
The mouse-down event that preceded the mouse-dragged event.

Discussion
This method has been deprecated in favor of `textView:draggedCell:inRect:event:atIndex:` (page 3888).

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- dragImage:at:offset:event:pasteboard:source:slideBack:` (page 3168) (NSView)
`- dragFile:fromRect:slideBack:event:` (page 3167) (NSView)

Declared In
NSTextView.h

textView:draggedCell:inRect:event:atIndex:

Sent when the user attempts to drag a cell.

```
- (void)textView:(NSTextView *)aTextView draggedCell:(id < NSTextAttachmentCell
    >)cell inRect:(NSRect)rect event:(NSEvent *)event atIndex:(NSUInteger)charIndex
```

Parameters

aTextView

The text view sending the message.

cell

The cell being dragged.

aRect

The rectangle from which the cell was dragged.

theEvent

The mouse-down event that preceded the mouse-dragged event.

charIndex

The character position where the mouse button was clicked.

Discussion

The delegate can use this message as its cue to initiate a dragging operation.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- dragImage:at:offset:event:pasteboard:source:slideBack: (page 3168) (NSView)
- dragFile:fromRect:slideBack:event: (page 3167) (NSView)

Declared In

NSTextView.h

textView:menu:forEvent:atIndex:

Allows delegate to control the context menu returned by the text view.

```
- (NSMenu *)textView:(NSTextView *)view menu:(NSMenu *)menu forEvent:(NSEvent
    *)event atIndex:(NSUInteger)charIndex
```

Parameters

view

The text view sending the message.

menu

The proposed contextual menu.

event

The mouse-down event that initiated the contextual menu's display.

charIndex

The character position where the mouse button was clicked.

Return Value
A menu to use as the contextual menu. You can return *menu* unaltered, or you can return a customized menu.

Discussion
This method allows the delegate to control the context menu returned by `menuForEvent:` (page 3189).

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTextView.h`

textView:shouldChangeTextInRange:replacementString:

Sent when a text view needs to determine if text in a specified range should be changed.

```
- (BOOL)textView:(NSTextView *)aTextView
    shouldChangeTextInRange:(NSRange)affectedCharRange replacementString:(NSString
     *)replacementString
```

Parameters

aTextView
The text view sending the message. This is the first text view in a series shared by a layout manager, not necessarily the text view displaying the selected text.

affectedCharRange
The range of characters to be replaced.

replacementString
The characters that will replace the characters in *affectedCharRange*; `nil` if only text attributes are being changed.

Return Value
`YES` to allow the replacement, or `NO` to reject the change.

Discussion
If a delegate implements this method and not its multiple-selection replacement, `textView:shouldChangeTextInRanges:replacementStrings:` (page 3889), it is called with an appropriate range and string. If a delegate implements the new method, then this one is ignored.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTextView.h`

textView:shouldChangeTextInRanges:replacementStrings:

Sent when a text view needs to determine if text in an array of specified ranges should be changed.

```
- (BOOL)textView:(NSTextView *)textView shouldChangeTextInRanges:(NSArray
    *)affectedRanges replacementStrings:(NSArray *)replacementStrings
```

Parameters

textView

The text view sending the message. This is the first text view in a series shared by a layout manager, not necessarily the text view displaying the selected text.

affectedRanges

The array of ranges of characters to be replaced. This array must be a non-nil, non-empty array of objects responding to the NSValue `rangeValue` method, and in addition its elements must be sorted, non-overlapping, non-contiguous, and (except for the case of a single range) have non-zero-length.

replacementStrings

The array of strings that will replace the characters in *affectedRanges*, one string for each range; `nil` if only text attributes are being changed.

Return Value

`YES` to allow the replacement, or `NO` to reject the change.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTextView.h`

textView:shouldChangeTypingAttributes:toAttributes:

Sent when the typing attributes are changed.

```
- (NSDictionary *)textView:(NSTextView *)textView
    shouldChangeTypingAttributes:(NSDictionary *)oldTypingAttributes
    toAttributes:(NSDictionary *)newTypingAttributes
```

Parameters

textView

The text view sending the message.

oldTypingAttributes

The old typing attributes.

newTypingAttributes

The proposed typing attributes.

Return Value

The actual new typing attributes.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTextView.h`

textView:shouldSetSpellingState:range:

Sent when the spelling state is changed.

```
- (NSInteger)textView:(NSTextView *)textView shouldSetSpellingState:(NSInteger)value
    range:(NSRange)affectedCharRange
```

Parameters

textView

The text view sending the message.

value

The proposed spelling state value to set. Possible values, for the temporary attribute on the layout manager using the key `NSSpellingStateAttributeName`, are:

`NSSpellingStateSpellingFlag` (page 287) to highlight spelling issues.

`NSSpellingStateGrammarFlag` (page 287) to highlight grammar issues.

affectedCharRange

The character range over which to set the given spelling state.

Return Value
The actual spelling state to set.

Discussion
Delegate only. Allows delegate to control the setting of spelling and grammar indicators.

Availability
Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- setSpellingState:range:` (page 2945) (NSTextView)

Declared In
`NSTextView.h`

textView:willChangeSelectionFromCharacterRange:toCharacterRange:

Returns the actual range to select.

```
- (NSRange)textView:(NSTextView *)aTextView
    willChangeSelectionFromCharacterRange:(NSRange)oldSelectedCharRange
    toCharacterRange:(NSRange)newSelectedCharRange
```

Parameters

aTextView

The text view sending the message. This is the first text view in a series shared by a layout manager, not necessarily the text view displaying the selected text.

oldSelectedCharRange

The original range of the selection.

newSelectedCharRange

The proposed character range for the new selection.

Return Value
The actual character range for the new selection.

Discussion
This method is invoked before a text view finishes changing the selection—that is, when the last argument to a `setSelectedRange:affinity:stillSelecting:` (page 2941) message is `NO`.

Special Considerations

In Mac OS X version 10.4 and later, if a delegate implements this delegate method and not its multiple-selection replacement, `textView:willChangeSelectionFromCharacterRanges:toCharacterRanges:` (page 3892), then multiple selection is effectively disallowed; attempts to set the selected ranges call the old delegate method with the first subrange, and afterwards only a single selected range is set.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- textView:willChangeSelectionFromCharacterRanges:toCharacterRanges:` (page 3892)

Declared In
`NSTextView.h`

textView:willChangeSelectionFromCharacterRanges:toCharacterRanges:

Returns the actual character ranges to select.

```
- (NSArray *)textView:(NSTextView *)aTextView
    willChangeSelectionFromCharacterRanges:(NSArray *)oldSelectedCharRanges
    toCharacterRanges:(NSArray *)newSelectedCharRanges
```

Parameters

aTextView
The text view sending the message. This is the first text view in a series shared by a layout manager, not necessarily the text view displaying the selected text.

oldSelectedCharRanges
An array containing the original ranges of the selection. This must be a non-`nil`, non-empty array of objects responding to the NSValue `rangeValue` method, and in addition its elements must be sorted, non-overlapping, non-contiguous, and (except for the case of a single range) have non-zero-length.

newSelectedCharRanges
An array containing the proposed character ranges for the new selection. This must be a non-`nil`, non-empty array of objects responding to the NSValue `rangeValue` method, and in addition its elements must be sorted, non-overlapping, non-contiguous, and (except for the case of a single range) have non-zero-length.

Return Value
An array containing the actual character ranges for the new selection.

Discussion
Invoked before an NSTextView finishes changing the selection—that is, when the last argument to a `setSelectedRange:affinity:stillSelecting:` (page 2941) or `setSelectedRanges:affinity:stillSelecting:` (page 2943) message is `NO`.

If a delegate implements both this method and `textView:willChangeSelectionFromCharacterRange:toCharacterRange:` (page 3891), then the latter is ignored.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTextView.h

textView:willCheckTextInRange:options:types:

Invoked to allow the delegate to modify the text checking process before it occurs.

```
- (NSDictionary *)textView:(NSTextView *)view willCheckTextInRange:(NSRange)range
    options:(NSDictionary *)options types:(NSTextCheckingTypes *)checkingTypes
```

Parameters

view
The text view sending the message.

range
The range to be checked.

options
A dictionary of values used during the checking process to perform. See Spell Checking Option Dictionary Keys (page 2537) for the supported values.

checkingTypes
The type of checking to be performed, passed by-reference. The possible constants are listed in NSTextCheckingTypes and can be combined using the C bit-wise OR operator to perform multiple checks at the same time.

You can change this parameter to alter the types of checking to be performed.

Return Value
A dictionary containing an alternative to the options *dictionary*.

Discussion
Invoked by checkTextInRange:types:options: (page 2886), this method allows control over text checking *options*s (via the return value) or types (by modifying the flags pointed to by the inout parameter *checkingTypes*)

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSTextView.h

textView:willDisplayToolTip:forCharacterAtIndex:

Returns the actual tooltip to display.

```
- (NSString *)textView:(NSTextView *)textView willDisplayToolTip:(NSString *)tooltip
    forCharacterAtIndex:(NSUInteger)characterIndex
```

Parameters

textView
The text view sending the message.

tooltip
The proposed tooltip to display.

characterIndex
The location in *textView*.

Return Value
The actual tooltip to display, or `nil` to suppress display of the tooltip.

Discussion
The tooltip string is the value of the `NSToolTipAttributeName` (page 274) attribute at *characterIndex*.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTextView.h`

textView:writablePasteboardTypesForCell:atIndex:

Returns the writable pasteboard types for a given cell.

```
- (NSArray *)textView:(NSTextView *)aTextView writablePasteboardTypesForCell:(id
    < NSTextAttachmentCell >)cell atIndex:(NSUInteger)charIndex
```

Parameters

aTextView
The text view sending the message.

cell
The cell in question.

charIndex
The character index in the text view that was clicked.

Return Value
An array of types that can be written to the pasteboard for *cell*.

Discussion
This method is invoked after the user clicks *cell* at the specified *charIndex* location in *aTextView*. If the `textView:draggedCell:inRect:event:atIndex:` (page 3888) is not used, this method and `textView:writeCell:atIndex:toPasteboard:type:` (page 3895) allow *aTextView* to take care of attachment dragging and pasting, with the delegate responsible only for writing the attachment to the pasteboard.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTextView.h`

textView:writeCell:atIndex:toPasteboard:type:

Returns whether data of the specified type for the given cell could be written to the specified pasteboard.

```
- (BOOL)textView:(NSTextView *)aTextView writeCell:(id < NSTextAttachmentCell >)cell
    atIndex:(NSUInteger)charIndex toPasteboard:(NSPasteboard *)pboard type:(NSString
    *)type
```

Parameters

aTextView
The text view sending the message.

cell
The cell whose contents should be written to the pasteboard.

charIndex
The index at which the cell was accessed.

pboard
The pasteboard to which the cell's contents should be written.

type
The type of data that should be written.

Return Value

`YES` if the write succeeded, `NO` otherwise.

Discussion

The receiver should attempt to write the *cell* to *pboard* with the given *type*, and return success or failure.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTextView.h`

textViewDidChangeSelection:

Sent when the selection changes in the text view.

```
- (void)textViewDidChangeSelection:(NSNotification *)aNotification
```

Parameters

aNotification
A notification named `NSTextViewDidChangeSelectionNotification` (page 2974).

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTextView.h`

textViewDidChangeTypingAttributes:

Sent when a text view's typing attributes change.

```
- (void)textViewDidChangeTypingAttributes:(NSNotification *)aNotification
```

Parameters

aNotification

A notification named `NSTextViewDidChangeTypingAttributesNotification` (page 2975).

Availability

Available in Mac OS X v10.3 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTextView.h`

undoManagerForTextView:

Returns the undo manager for the specified text view.

```
- (NSUndoManager *)undoManagerForTextView:(NSTextView *)aTextView
```

Parameters

aTextView

The text view whose undo manager should be returned.

Return Value

The undo manager for *aTextView*.

Discussion

This method provides the flexibility to return a custom undo manager for the text view. Although `NSTextView` implements undo and redo for changes to text, applications may need a custom undo manager to handle interactions between changes to text and changes to other items in the application.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTextView.h`

NSTokenFieldCellDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSTokenFieldCell.h
Companion guide	Token Field Programming Guide

Overview

The `NSTokenFieldCellDelegate` protocol defines the optional methods implemented by delegates of `NSTokenFieldCell` objects.

Tasks

Displaying Tokenized Strings

`- tokenFieldCell:displayStringForRepresentedObject:` (page 3899) Available in Mac OS X v10.0 through Mac OS X v10.1 Available in Mac OS X v10.5 through Mac OS X v10.5

Allows the delegate to provide a string to be displayed as a proxy for the represented object.

`- tokenFieldCell:styleForRepresentedObject:` (page 3903) Available in Mac OS X v10.0 through Mac OS X v10.1 Available in Mac OS X v10.5 through Mac OS X v10.5

Allows the delegate to return the token style for editing the specified represented object.

Editing a Tokenized Strings

`- tokenFieldCell:editingStringForRepresentedObject:` (page 3899)

Allows the delegate to provide a string to be edited as a proxy for the represented object.

`- tokenFieldCell:representedObjectForEditingString:` (page 3901)

Allows the delegate to provide a represented object for the string being edited.

`- tokenFieldCell:shouldAddObjects:atIndex:` (page 3902) Available in Mac OS X v10.5 through Mac OS X v10.5

Allows the delegate to validate the tokens to be added to the receiver at a given index.

- tokenFieldCell:completionsForSubstring:indexOfToken:indexOfSelectedItem: (page 3898)
Available in Mac OS X v10.0 through Mac OS X v10.1 Available in Mac OS X v10.5 through Mac OS X v10.5
Allows the delegate to provide an array of appropriate completions for the contents of the receiver.

Reading To and Writing From the Pasteboard

- tokenFieldCell:readFromPasteboard: (page 3901)
Allows the delegate to return an array of objects representing the data read from *pboard*.
- tokenFieldCell:writeRepresentedObjects:toPasteboard: (page 3903)
Allows the delegate the opportunity to write custom pasteboard types to the pasteboard for the represented objects in *objects*.

Managing Menus for Represented Objects

- tokenFieldCell:hasMenuForRepresentedObject: (page 3900)
Allows the delegate to specify whether the represented object provides a menu.
- tokenFieldCell:menuForRepresentedObject: (page 3900)
Allows the delegate to provide a menu for the specified represented object.

Instance Methods

tokenFieldCell:completionsForSubstring:indexOfToken:indexOfSelectedItem:

Allows the delegate to provide an array of appropriate completions for the contents of the receiver.

```
- (NSArray *)tokenFieldCell:(NSTokenFieldCell *)tokenFieldCell
    completionsForSubstring:(NSString *)substring indexOfToken:(NSInteger)tokenIndex
     indexOfSelectedItem:(NSInteger *)selectedIndex
```

Parameters

tokenFieldCell
The token field cell that sent the message.

substring
The partial string that is to be completed.

tokenIndex
The index of the token being edited.

selectedIndex
Optionally, you can return by-reference an index into the returned array that specifies which of the completions should be initially selected. If none are to be selected, return by reference `-1`.

Return Value
An array of strings that are possible completions.

Discussion
If the delegate does not implement this method, no completions are provided.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTokenFieldCell.h

tokenFieldCell:displayStringForRepresentedObject:

Allows the delegate to provide a string to be displayed as a proxy for the represented object.

```
- (NSString *)tokenFieldCell:(NSTokenFieldCell *)tokenFieldCell
    displayStringForRepresentedObject:(id)representedObject
```

Parameters

tokenFieldCell
The token field cell that sent the message.

representedObject
A represented object of the token field cell.

Return Value
The string to be used as a proxy for `representedObject`. If you return `nil` or do not implement this method, then *representedObject* is displayed as the string.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTokenFieldCell.h

tokenFieldCell:editingStringForRepresentedObject:

Allows the delegate to provide a string to be edited as a proxy for the represented object.

```
- (NSString *)tokenFieldCell:(NSTokenFieldCell *)tokenFieldCell
    editingStringForRepresentedObject:(id)representedObject
```

Parameters

tokenFieldCell
The token field cell that sent the message.

representedObject
A represented object of the token field.

Return Value
A string that's an editable proxy of the represented object, or `nil` if the token should not be editable.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- tokenFieldCell:representedObjectForEditingString: (page 3901)

Declared In
NSTokenFieldCell.h

tokenFieldCell:hasMenuForRepresentedObject:

Allows the delegate to specify whether the represented object provides a menu.

```
- (BOOL)tokenFieldCell:(NSTokenFieldCell *)tokenFieldCell
    hasMenuForRepresentedObject:(id)representedObject
```

Parameters

tokenFieldCell
The token field cell that sent the message.

representedObject
A represented object of the token field.

Return Value
YES if the represented object has a menu, NO otherwise.

Discussion
By default tokens have no menus.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- tokenFieldCell:menuForRepresentedObject: (page 3900)

Declared In
NSTokenFieldCell.h

tokenFieldCell:menuForRepresentedObject:

Allows the delegate to provide a menu for the specified represented object.

```
- (NSMenu *)tokenFieldCell:(NSTokenFieldCell *)tokenFieldCell
    menuForRepresentedObject:(id)representedObject
```

Parameters

tokenFieldCell
The token field cell that sent the message.

representedObject
A represented object of the token field.

Return Value
The menu associated with the represented object.

Discussion
The returned menu should be autoreleased. By default tokens in a token field cell do not return menus.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `tokenFieldCell:hasMenuForRepresentedObject:` (page 3900)

Declared In
`NSTokenFieldCell.h`

tokenFieldCell:readFromPasteboard:

Allows the delegate to return an array of objects representing the data read from *pboard*.

```
- (NSArray *)tokenFieldCell:(NSTokenFieldCell *)tokenFieldCell
    readFromPasteboard:(NSPasteboard *)pboard
```

Parameters

tokenFieldCell
The token field cell that sent the message.

pboard
The pasteboard from which to read the represented objects.

Return Value
An array of represented objects created from the pasteboard data.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `tokenFieldCell:writeRepresentedObjects:toPasteboard:` (page 3903)

Declared In
`NSTokenFieldCell.h`

tokenFieldCell:representedObjectForEditingString:

Allows the delegate to provide a represented object for the string being edited.

```
- (id)tokenFieldCell:(NSTokenFieldCell *)tokenFieldCell
    representedObjectForEditingString:(NSString *)editingString
```

Parameters

tokenFieldCell
The token field cell that sent the message.

editingString
The edited string representation of a represented object.

Return Value
A represented object that is displayed rather than the editing string.

Discussion
If your application uses some object other than an `NSString` for their represented objects, you should return a new, autoreleased instance of that object from this method.

> **Note:** In Mac OS X v10.4, `NSTokenField` trims whitespace around tokens but it does not trim whitespace in Mac OS X versions 10.5.0 and 10.5.1. In Mac OS X v10.5.2, you get whitespace-trimming behavior by either linking against the v10.4 binary or linking against the v10.5 binary and *not* implementing the this method. If you do not want the whitespace-trimming behavior, link against the v10.5 binary and implement this method, returning the editing string if you have no represented object.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- tokenFieldCell:editingStringForRepresentedObject:` (page 3899)

Declared In
`NSTokenFieldCell.h`

tokenFieldCell:shouldAddObjects:atIndex:

Allows the delegate to validate the tokens to be added to the receiver at a given index.

```
- (NSArray *)tokenFieldCell:(NSTokenFieldCell *)tokenFieldCell
    shouldAddObjects:(NSArray *)tokens atIndex:(NSUInteger)index
```

Parameters

`tokenFieldCell`
The token field cell that sent the message.

`tokens`
An array of tokens to be inserted in the receiver at `index`.

`index`
The index of the receiver in which the array of tokens to be validated (`tokens`) will be inserted.

Return Value
An array of validated tokens.

Discussion
The delegate can return the array unchanged or return a modified array of tokens. To reject the add completely, return an empty array. Returning `nil` causes an error.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTokenFieldCell.h`

tokenFieldCell:styleForRepresentedObject:

Allows the delegate to return the token style for editing the specified represented object.

```
- (NSTokenStyle)tokenFieldCell:(NSTokenFieldCell *)tokenFieldCell
    styleForRepresentedObject:(id)representedObject
```

Parameters

tokenFieldCell

The token field cell that sent the message.

representedObject

A represented object of the token field cell.

Return Value

The style that should be used to display the representedObject. Possible values are shown in `NSTokenStyle_Values`.

Discussion

If the delegate implements this method and returns an `NSTokenStyle` (page 2988) that differs from the style set by `setTokenStyle:` (page 2987), the value the delegate returns is preferred.

If the delegate does not implement this method, the token field cell's `tokenStyle` (page 2988) is used.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTokenFieldCell.h`

tokenFieldCell:writeRepresentedObjects:toPasteboard:

Allows the delegate the opportunity to write custom pasteboard types to the pasteboard for the represented objects in *objects*.

```
- (BOOL)tokenFieldCell:(NSTokenFieldCell *)tokenFieldCell
    writeRepresentedObjects:(NSArray *)objects toPasteboard:(NSPasteboard *)pboard
```

Parameters

tokenFieldCell

The token field cell that sent the message.

objects

An array of represented objects associated with the token field cell.

pboard

The pasteboard to which to write the represented objects.

Return Value

`YES` if the delegate writes the represented objects to the pasteboard, `NO` otherwise. If `NO`, the token field writes the display strings to the `NSStringPboardType` (page 1909) pasteboard.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `tokenFieldCell:readFromPasteboard:` (page 3901)

Declared In
`NSTokenFieldCell.h`

NSTokenFieldDelegate Protocol Reference

Conforms to	NSTextFieldDelegate
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSTokenField.h
Companion guide	Token Field Programming Guide

Overview

The `NSTokenFieldDelegate` protocol defines the optional methods implemented by delegates of `NSTokenField` objects.

Tasks

Displaying Tokenized Strings

- `tokenField:displayStringForRepresentedObject:` (page 3907)
  Allows the delegate to provide a string to be displayed as a proxy for the given represented object.
- `tokenField:styleForRepresentedObject:` (page 3910)
  Allows the delegate to return the token style for editing the specified represented object.

Editing a Tokenized Strings

- `tokenField:completionsForSubstring:indexOfToken:indexOfSelectedItem:` (page 3906)
  Allows the delegate to provide an array of appropriate completions for the contents of the receiver.
- `tokenField:editingStringForRepresentedObject:` (page 3907)
  Allows the delegate to provide a string to be edited as a proxy for a represented object.
- `tokenField:representedObjectForEditingString:` (page 3909)
  Allows the delegate to provide a represented object for the given editing string.
- `tokenField:shouldAddObjects:atIndex:` (page 3910)
  Allows the delegate to validate the tokens to be added to the receiver at a particular location.

Reading To and Writing From the Pasteboard

- `tokenField:readFromPasteboard:` (page 3909)
  Allows the delegate to return an array of objects representing the data read from the specified pasteboard.
- `tokenField:writeRepresentedObjects:toPasteboard:` (page 3911)
  Sent so the delegate can write represented objects to the pasteboard corresponding to a given array of display strings.

Managing Menus for Represented Objects

- `tokenField:hasMenuForRepresentedObject:` (page 3908)
  Allows the delegate to specify whether the given represented object provides a menu.
- `tokenField:menuForRepresentedObject:` (page 3908)
  Allows the delegate to provide a menu for the specified represented object.

Instance Methods

tokenField:completionsForSubstring:indexOfToken:indexOfSelectedItem:

Allows the delegate to provide an array of appropriate completions for the contents of the receiver.

```
- (NSArray *)tokenField:(NSTokenField *)tokenField completionsForSubstring:(NSString
    *)substring indexOfToken:(NSInteger)tokenIndex indexOfSelectedItem:(NSInteger
    *)selectedIndex
```

Parameters

tokenField
The token field where editing is occurring.

substring
The partial string that is to be completed.

tokenIndex
The index of the token being edited.

selectedIndex
Optionally, you can return by-reference an index into the returned array that specifies which of the completions should be initially selected. If none are to be selected, return by reference `-1`.

Return Value
An array of strings that are possible completions.

Discussion
If the delegate does not implement this method, no completions are provided.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTokenField.h

tokenField:displayStringForRepresentedObject:

Allows the delegate to provide a string to be displayed as a proxy for the given represented object.

```
- (NSString *)tokenField:(NSTokenField *)tokenField
    displayStringForRepresentedObject:(id)representedObject
```

Parameters

tokenField
The token field that sent the message.

representedObject
A represented object of the token field.

Return Value
The string to be used as a proxy for `representedObject`. If you return `nil` or do not implement this method, then *representedObject* is displayed as the string.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSTokenField.h

tokenField:editingStringForRepresentedObject:

Allows the delegate to provide a string to be edited as a proxy for a represented object.

```
- (NSString *)tokenField:(NSTokenField *)tokenField
    editingStringForRepresentedObject:(id)representedObject
```

Parameters

tokenField
The token field that sent the message.

representedObject
A represented object of the token field.

Return Value
A string that's an editable proxy of the represented object, or `nil` if the token should not be editable.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `tokenField:representedObjectForEditingString:` (page 3909)

Declared In
NSTokenField.h

tokenField:hasMenuForRepresentedObject:

Allows the delegate to specify whether the given represented object provides a menu.

```
- (BOOL)tokenField:(NSTokenField *)tokenField
    hasMenuForRepresentedObject:(id)representedObject
```

Parameters

tokenField

The token field that sent the message.

representedObject

A represented object of the token field.

Return Value

`YES` if the represented object has a menu, `NO` otherwise.

Discussion

By default tokens in a token field have no menus.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

`- tokenField:menuForRepresentedObject:` (page 3908)

Declared In

`NSTokenField.h`

tokenField:menuForRepresentedObject:

Allows the delegate to provide a menu for the specified represented object.

```
- (NSMenu *)tokenField:(NSTokenField *)tokenField
    menuForRepresentedObject:(id)representedObject
```

Parameters

tokenField

The token field that sent the message.

representedObject

A represented object of the token field.

Return Value

The menu associated with the represented object.

Discussion

The returned menu should be autoreleased. By default tokens in a token field do not return menus.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

`- tokenField:hasMenuForRepresentedObject:` (page 3908)

Declared In
NSTokenField.h

tokenField:readFromPasteboard:

Allows the delegate to return an array of objects representing the data read from the specified pasteboard.

```
- (NSArray *)tokenField:(NSTokenField *)tokenField readFromPasteboard:(NSPasteboard
    *)pboard
```

Parameters

tokenField
The token field that sent the message.

pboard
The pasteboard from which to read the represented objects.

Return Value
An array of represented objects created from the pasteboard data.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- tokenField:writeRepresentedObjects:toPasteboard: (page 3911)

Declared In
NSTokenField.h

tokenField:representedObjectForEditingString:

Allows the delegate to provide a represented object for the given editing string.

```
- (id)tokenField:(NSTokenField *)tokenField
    representedObjectForEditingString:(NSString *)editingString
```

Parameters

tokenField
The token field that sent the message.

editingString
The edited string representation of a represented object.

Return Value
A represented object that is displayed rather than the editing string.

Discussion
If your application uses some object other than an NSString for their represented objects, you should return a new, autoreleased instance of that object from this method.

Note: In Mac OS X v10.4, `NSTokenField` trims whitespace around tokens but it does not trim whitespace in Mac OS X versions 10.5.0 and 10.5.1. In Mac OS X v10.5.2, you get whitespace-trimming behavior by either linking against the v10.4 binary or linking against the v10.5 binary and *not* implementing the this method. If you do not want the whitespace-trimming behavior, link against the v10.5 binary and implement this method, returning the editing string if you have no represented object.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- tokenField:editingStringForRepresentedObject:` (page 3907)

Declared In
`NSTokenField.h`

tokenField:shouldAddObjects:atIndex:

Allows the delegate to validate the tokens to be added to the receiver at a particular location.

```
- (NSArray *)tokenField:(NSTokenField *)tokenField shouldAddObjects:(NSArray *)tokens
    atIndex:(NSUInteger)index
```

Parameters

tokenField
The token field that sent the message.

tokens
An array of tokens to be inserted in the receiver at `index`.

index
The index of the receiver in which the array of tokens to be validated (`tokens`) will be inserted.

Return Value
An array of validated tokens.

Discussion
The delegate can return the array unchanged or return a modified array of tokens. To reject the add completely, return an empty array. Returning `nil` causes an error.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSTokenField.h`

tokenField:styleForRepresentedObject:

Allows the delegate to return the token style for editing the specified represented object.

```
- (NSTokenStyle)tokenField:(NSTokenField *)tokenField
    styleForRepresentedObject:(id)representedObject
```

Parameters

tokenField

The token field that sent the message.

representedObject

A represented object of the token field.

Return Value

The style that should be used to display the representedObject. Possible values are shown in `NSTokenStyle Values`.

Discussion

If the delegate implements this method and returns an `NSTokenStyle` (page 2988) that differs from the style set by `setTokenStyle:` (page 2981), the value the delegate returns is preferred.

If the delegate does not implement this method, the token field's `tokenStyle` (page 2982) is used.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSTokenField.h`

tokenField:writeRepresentedObjects:toPasteboard:

Sent so the delegate can write represented objects to the pasteboard corresponding to a given array of display strings.

```
- (BOOL)tokenField:(NSTokenField *)tokenField writeRepresentedObjects:(NSArray
    *)objects toPasteboard:(NSPasteboard *)pboard
```

Parameters

tokenField

The token field that sent the message.

objects

An array of represented objects associated with the token field.

pboard

The pasteboard to which to write the represented objects.

Return Value

`YES` if the delegate writes the represented objects to the pasteboard, `NO` otherwise. If `NO`, the token field writes the display strings to the `NSStringPboardType` (page 1909) pasteboard.

Availability

Available in Mac OS X v10.4 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

`- tokenField:readFromPasteboard:` (page 3909)

Declared In

`NSTokenField.h`

NSToolbarDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSToolbar.h
Companion guide	Toolbar Programming Topics for Cocoa
Related sample code	iSpend SimpleCocoaBrowser SimpleToolbar

Overview

The `NSToolbarDelegate` protocol defines the optional methods implemented by delegates of `NSToolbar` objects.

Tasks

Configuring a Toolbar

- `toolbar:itemForItemIdentifier:willBeInsertedIntoToolbar:` (page 3914)
  Sent to request a new toolbar item; returns a toolbar item of the identified kind for the specified toolbar.
- `toolbarAllowedItemIdentifiers:` (page 3915)
  Sent to discover the allowed item identifiers for a toolbar.
- `toolbarDefaultItemIdentifiers:` (page 3915)
  Sent to discover the default item identifiers for a toolbar.
- `toolbarSelectableItemIdentifiers:` (page 3916)
  Sent to discover the selectable item identifiers for a toolbar.

Responding to Additions and Deletions in the Toolbar

- `toolbarWillAddItem:` (page 3917)
  Sent just before a new item is added to a toolbar.
- `toolbarDidRemoveItem:` (page 3916)
  Sent just after an item has been removed from a toolbar.

Instance Methods

toolbar:itemForItemIdentifier:willBeInsertedIntoToolbar:

Sent to request a new toolbar item; returns a toolbar item of the identified kind for the specified toolbar.

```
- (NSToolbarItem *)toolbar:(NSToolbar *)toolbar itemForItemIdentifier:(NSString
    *)itemIdentifier willBeInsertedIntoToolbar:(BOOL)flag
```

Parameters

toolbar
  The toolbar for which the item is being requested.

itemIdentifier
  The identifier for the requested item.

flag
  `YES` if the item will be immediately inserted into the toolbar. If *flag* is `NO` the toolbar item is being requested for display in the toolbar customization sheet and should always be enabled or provide some other canonical representation. If you ignore this parameter the same toolbar item will be used in the toolbar and in the customization sheet.

Return Value
The toolbar item for the specified toolbar and identifier. Return `nil` to indicate that the identified kind of toolbar item is not supported. When an item is requested again, you may return the same `NSToolbarItem` object returned earlier or a different instance.

Discussion
Implement this method to create new toolbar item instances. This method is called lazily on behalf of a toolbar instance, which must be the sole owner of the toolbar item. A toolbar may ask again for a kind of toolbar item already supplied to it, in which case this method may return the same toolbar item it returned before or a different one. If your delegate services multiple toolbars, each attached to a different window, it is best to return a different item for each toolbar—an `NSToolbarItem` object can only be in one toolbar at a time.

If the item is a custom view item, the `NSView` object must be fully formed when the item is returned. Do not assume that the returned item is going to be added as an active item in the toolbar, as it could be that it will be used only in the customization palette. (The customization palette makes a copy of the returned item.)

Important: While this method is marked as `@optional` in the `NSToolbarDelegate` protocol , it must be implemented if the associated toolbar is created programatically. Toolbars created in Interface Builder can implement this method to augment functionality.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSToolbar.h`

toolbarAllowedItemIdentifiers:

Sent to discover the allowed item identifiers for a toolbar.

```
- (NSArray *)toolbarAllowedItemIdentifiers:(NSToolbar *)toolbar
```

Parameters
toolbar
The toolbar whose allowed item identifiers are to be returned.

Return Value
An array of toolbar item identifiers for *toolbar*, specifying the contents and the order of the items in the configuration palette.

Discussion
Every allowed item must be explicitly listed, even the standard ones. The identifiers returned should include all of those returned by `toolbarDefaultItemIdentifiers:` (page 3915).

Important: While this method is marked as `@optional` in the `NSToolbarDelegate` protocol , it must be implemented if the associated toolbar is created programatically. Toolbars created in Interface Builder can implement this method to augment functionality.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSToolbar.h`

toolbarDefaultItemIdentifiers:

Sent to discover the default item identifiers for a toolbar.

```
- (NSArray *)toolbarDefaultItemIdentifiers:(NSToolbar *)toolbar
```

Parameters
toolbar
The toolbar whose default item identifiers are to be returned.

Return Value
An array of toolbar item identifiers for *toolbar*, specifying the contents and the order of the items in the default toolbar configuration.

Discussion
During initialization of *toolbar*, this method is called only if a toolbar configuration for the identifier of *toolbar* is not found in the user preferences. This method is called during initialization of the toolbar customization palette.

Important: While this method is marked as `@optional` in the `NSToolbarDelegate` protocol , it must be implemented if the associated toolbar is created programatically. Toolbars created in Interface Builder can implement this method to augment functionality.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSToolbar.h`

toolbarDidRemoveItem:

Sent just after an item has been removed from a toolbar.

```
- (void)toolbarDidRemoveItem:(NSNotification *)notification
```

Parameters
notification
A notification named `NSToolbarDidRemoveItemNotification` (page 3005).

Discussion
This method allows you to remove information related to the item that may have been cached.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSToolbar.h`

toolbarSelectableItemIdentifiers:

Sent to discover the selectable item identifiers for a toolbar.

```
- (NSArray *)toolbarSelectableItemIdentifiers:(NSToolbar *)toolbar
```

Parameters
toolbar
The toolbar whose selectable item identifiers are to be returned.

Return Value
An array of item identifiers that should indicate selection in the specified *toolbar*.

Discussion
Toolbars that need to indicate item selection should return an array containing the identifiers of the selectable toolbar items.

If implemented, *toolbar* will display the currently selected item with a visual highlight. Clicking on an item whose identifier is selectable will automatically update the toolbar's selected item identifier, when possible. Clicking an item whose identifier is not selectable will not update the toolbar's selected item identifier.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `setSelectedItemIdentifier:` (page 3000) (NSToolbar)

Declared In
`NSToolbar.h`

toolbarWillAddItem:

Sent just before a new item is added to a toolbar.

```
- (void)toolbarWillAddItem:(NSNotification *)notification
```

Parameters
notification
A notification named `NSToolbarWillAddItemNotification` (page 3005).

Discussion
If you need to cache a reference to a toolbar item or need to set up some initial state before a toolbar item is added, this is where to do it.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSToolbar.h`

NSToolbarItemValidation Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSToolbarItem.h
Companion guide	Toolbar Programming Topics for Cocoa

Overview

A toolbar item with a valid target and action is enabled by default. To allow a toolbar item to be disabled in certain situations, a toolbar item's target can implement the `validateToolbarItem:` (page 3919) method.

Note: NSToolbarItem's `validate` (page 3024) method calls this method only if the item's target has a valid action defined on its target and if the item is not a custom view item. If you want to validate a custom view item, then you have to subclass NSToolbarItem and override `validate` (page 3024).

Tasks

Validating Toolbar Items

`- validateToolbarItem:` (page 3919)
If this method is implemented and returns `NO`, NSToolbar will disable *`theItem`*; returning `YES` causes *`theItem`* to be enabled.

Instance Methods

validateToolbarItem:

If this method is implemented and returns `NO`, NSToolbar will disable *`theItem`*; returning `YES` causes *`theItem`* to be enabled.

```
- (BOOL)validateToolbarItem:(NSToolbarItem *)theItem
```

Discussion
NSToolbar only calls this method for image items.

Note: `validateToolbarItem:` is called very frequently, so it must be efficient.

If the receiver is the `target` for the actions of multiple toolbar items, it's necessary to determine which toolbar item *`theItem`* refers to by testing the `itemIdentifier`.

```
-(BOOL)validateToolbarItem:(NSToolbarItem *)toolbarItem
{
    BOOL enable = NO;
    if ([[toolbarItem itemIdentifier] isEqual:SaveDocToolbarItemIdentifier]) {
        // We will return YES (enable the save item)
        // only when the document is dirty and needs saving
        enable = [self isDocumentEdited];
    } else if ([[toolbarItem itemIdentifier]
isEqual:NSToolbarPrintItemIdentifier]) {
        // always enable print for this window
        enable = YES;
    }
    return enable;
}
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `validateVisibleItems` (page 3003) (NSToolbar)
- `validate` (page 3024) (NSToolbarItem)
- `target` (page 3023) (NSToolbarItem)
- `action` (page 3010) (NSToolbarItem)

Declared In
`NSToolbarItem.h`

NSToolTipOwner Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSView.h
Companion guide	Online Help

Overview

The NSToolTipOwner informal protocol declares a method that allows an object to dynamically provide the text to a tool tip. If the tool tip object does not implement this method, the NSObject protocol `description` method is invoked instead.

Tasks

Obtaining Tool Tip Strings

`- view:stringForToolTip:point:userData:` (page 3921)
Returns the tool tip string to be displayed due to the cursor pausing at location *point* within the tool tip rectangle identified by *tag* in the view *view*.

Instance Methods

view:stringForToolTip:point:userData:

Returns the tool tip string to be displayed due to the cursor pausing at location *point* within the tool tip rectangle identified by *tag* in the view *view*.

```
- (NSString *)view:(NSView *)view stringForToolTip:(NSToolTipTag)tag
    point:(NSPoint)point userData:(void *)userData
```

Discussion
userData is additional information provided by the creator of the tool tip rectangle.

Availability
Available in Mac OS X v10.0 and later.

See Also
- addToolTipRect:owner:userData: (page 3141) (NSView)

Declared In
NSView.h

NSUserInterfaceValidations Protocol Reference

Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSUserInterfaceValidation.h
Companion guide	User Interface Validation

Overview

The `NSUserInterfaceValidations` protocol works with the `NSValidatedUserInterfaceItem` protocol to allow the target of a user interface element such as a menu item or a toolbar item to decide whether or not the user interface element should be enabled.

Your custom classes should adopt this protocol if an instance may be the target of a user interface element and need to conditionally enable or disable the element based on the current state of the instance. For more details, read *User Interface Validation*.

Tasks

Validating User Interface Items

`- validateUserInterfaceItem:` (page 3923) *required method*
Returns a Boolean value that indicates whether the sender should be enabled. (required)

Instance Methods

validateUserInterfaceItem:

Returns a Boolean value that indicates whether the sender should be enabled. (required)

```
- (BOOL)validateUserInterfaceItem:(id < NSValidatedUserInterfaceItem >)anItem
```

Parameters

anItem

The user interface item to validate. You can send *anItem* the `action` (page 3925) and `tag` (page 3926) messages.

Return Value

`YES` if the user interface item should be enabled, otherwise `NO`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSUserInterfaceValidation.h`

NSValidatedUserInterfaceItem Protocol Reference

Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	AppKit/NSUserInterfaceValidation.h
Companion guide	User Interface Validation

Overview

The `NSValidatedUserInterfaceItem` protocol works with the `NSUserInterfaceValidations` protocol to enable or disable a control automatically, depending on whether any responder in the responder chain can handle the control's action method. The `NSMenuItem` and `NSToolbarItem` classes implement this protocol.

By conforming to this protocol, your control can participate in this validation mechanism. To validate a control, the application calls `validateUserInterfaceItem:` for each item in the responder chain, starting with the first responder. If no responder returns `YES`, the item is disabled. For example, a menu item that sends the `copy:` action message would disable itself if no responder in the responder chain can be copied.

Tasks

Getting Information About a User Interface Item

- `action` (page 3925) *required method*
  Returns the selector of the receiver's action method. (required)
- `tag` (page 3926) *required method*
  Returns the receiver's tag integer. (required)

Instance Methods

action

Returns the selector of the receiver's action method. (required)

```
- (SEL)action
```

Return Value
The selector of the receiver's action method.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSUserInterfaceValidation.h`

tag

Returns the receiver’s tag integer. (required)

```
- (NSInteger)tag
```

Return Value
The receiver's tag.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSUserInterfaceValidation.h`

NSWindowDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.6 and later.
Declared in	AppKit/NSWindow.h
Companion guide	Window Programming Guide
Related sample code	UIElementInspector

Overview

The `NSWindowDelegate` protocol defines the methods that a delegate of `NSWindow` should implement. All methods in this protocol are optional.

By implementing these methods, the delegate may respond to window resizing, moving, exposing, minimizing, and a number of other window events.

Tasks

Managing Sheets

- `window:willPositionSheet:usingRect:` (page 3931)
  Tells the delegate that the window is about to show a sheet at the specified location, giving it the opportunity to return a custom location for the attachment of the sheet to the window.
- `windowWillBeginSheet:` (page 3938)
  Notifies the delegate that the window is about to open a sheet.
- `windowDidEndSheet:` (page 3934)
  Tells the delegate that the window has closed a sheet.

Sizing Windows

- `windowWillUseStandardFrame:defaultFrame:` (page 3942)
  Invoked by `NSWindow`'s `zoom:` (page 3409) method while determining the frame a window may be zoomed to.
- `windowShouldZoom:toFrame:` (page 3938)
  Asks the delegate whether the specified window should zoom to the specified frame.
- `windowWillResize:toSize:` (page 3940)
  Tells the delegate that the window is being resized (whether by the user or through one of the `setFrame...` methods other than `setFrame:display:` (page 3381)).
- `windowDidResize:` (page 3936)
  Informs the delegate that the window has been resized.
- `windowWillStartLiveResize:` (page 3941)
  Informs the delegate that the window is about to be live resized.
- `windowDidEndLiveResize:` (page 3934)
  Informs the delegate that a live resize operation on the window has ended.

Managing Key Status

- `windowDidBecomeKey:` (page 3932)
  Informs the delegate that the window has become the key window.
- `windowDidResignKey:` (page 3936)
  Informs the delegate that the window has resigned key window status.

Managing Main Status

- `windowDidBecomeMain:` (page 3932)
  Informs the delegate that the window has become main.
- `windowDidResignMain:` (page 3936)
  Informs the delegate that the window has resigned main window status.

Managing Field Editors

- `windowWillReturnFieldEditor:toObject:` (page 3940)
  Tells the delegate that the field editor for a text-displaying object has been requested.

Updating Windows

- `windowDidUpdate:` (page 3937)
  Tells the delegate that the window received an `update` (page 3405) message.

Exposing Windows

- `windowDidExpose:` (page 3935)
  Tells the delegate that the window has been exposed.

Dragging Windows

- `window:shouldDragDocumentWithEvent:from:withPasteboard:` (page 3930)
  Asks the delegate whether a user can drag the document icon from the window's title bar.

Getting the Undo Manager

- `windowWillReturnUndoManager:` (page 3941)
  Tells the delegate that the window's undo manager has been requested. Returns the appropriate undo manager for the window.

Managing Titles

- `window:shouldPopUpDocumentPathMenu:` (page 3931)
  Asks the delegate whether the window displays the title pop-up menu in response to a Command-click or Control-click on its title.

Moving Windows

- `windowWillMove:` (page 3939)
  Tells the delegate that the window is about to move.
- `windowDidMove:` (page 3935)
  Tells the delegate that the window has moved.
- `windowDidChangeScreen:` (page 3933)
  Tells the delegate that the window has changed screens.
- `windowDidChangeScreenProfile:` (page 3933)
  Tells the delegate that the window has changed screen display profiles.

Closing Windows

- `windowShouldClose:` (page 3937)
  Tells the delegate that the user has attempted to close a window or the window has received a `performClose:` (page 3354) message.
- `windowWillClose:` (page 3939)
  Tells the delegate that the window is about to close.

Minimizing Windows

- `windowWillMiniaturize:` (page 3939)
  Tells the delegate that the window is about to be minimized.
- `windowDidMiniaturize:` (page 3935)
  Tells the delegate that the window has been minimized.
- `windowDidDeminiaturize:` (page 3933)
  Tells the delegate that the window has been deminimized.

Instance Methods

window:shouldDragDocumentWithEvent:from:withPasteboard:

Asks the delegate whether a user can drag the document icon from the window's title bar.

```
- (BOOL)window:(NSWindow *)window
    shouldDragDocumentWithEvent:(NSEvent *)event
    from:(NSPoint)dragImageLocation
    withPasteboard:(NSPasteboard *)pasteboard
```

Parameters

window
The window containing the document icon the user wants to drag.

event
The left-mouse down event that triggered the dragging operation.

dragImageLocation
The location at which the user started the dragging operation.

pasteboard
The pasteboard containing the contents of the document, which the delegate can modify.

Return Value
`YES` to allow the drag to proceed; `NO` to prevent it.

Discussion
To implement its own dragging process, the delegate can perform the dragging operation and return `NO`.

Availability
Available in Mac OS X v10.5 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- `representedURL` (page 3359)

Declared In
`NSWindow.h`

window:shouldPopUpDocumentPathMenu:

Asks the delegate whether the window displays the title pop-up menu in response to a Command-click or Control-click on its title.

```
- (BOOL)window:(NSWindow *)window
    shouldPopUpDocumentPathMenu:(NSMenu *)menu
```

Parameters

window

The window whose title the user Command-clicked or Control-clicked.

menu

The menu the window will display, if allowed. By default, its items are the path components of the file represented by *window*.

Return Value

`YES` to allow the display of the title pop-up menu; `NO` to prevent it.

Availability

Available in Mac OS X v10.5 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

See Also

- representedURL (page 3359)

Declared In

NSWindow.h

window:willPositionSheet:usingRect:

Tells the delegate that the window is about to show a sheet at the specified location, giving it the opportunity to return a custom location for the attachment of the sheet to the window.

```
- (NSRect)window:(NSWindow *)window
    willPositionSheet:(NSWindow *)sheet
    usingRect:(NSRect)rect
```

Parameters

window

The window containing the sheet to be animated.

sheet

The sheet to be shown.

rect

The default sheet location, just under the title bar of the window, aligned with the left and right edges of the window.

Return Value

The custom location specified.

Discussion

This method is also invoked whenever the user resizes *window* while *sheet* is attached.

This method is useful in many situations. If your window has a toolbar, for example, you can specify a location for the sheet that is just below it. If you want the sheet associated with a certain control or view, you could position the sheet so that it appears to originate from the object (through animation) or is positioned next to it.

Neither the *rect* parameter nor the returned `NSRect` value define the boundary of the sheet. They indicate where the top-left edge of the sheet is attached to the window. The origin is expressed in window coordinates; the default `origin.y` value is the height of the content view and the default `origin.x` value is 0. The `size.width` value indicates the width and behavior of the initial animation; if `size.width` is narrower than the sheet, the sheet genies out from the specified location, and if `size.width` is wider than the sheet, the sheet slides out. You cannot affect the size of the sheet through the `size.width` and `size.height` fields. It is recommended that you specify zero for the `size.height` value as this field may have additional meaning in a future release.

Availability
Available in Mac OS X v10.3 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSWindow.h`

windowDidBecomeKey:

Informs the delegate that the window has become the key window.

```
- (void)windowDidBecomeKey:(NSNotification *)notification
```

Parameters
notification
A notification named `NSWindowDidBecomeKeyNotification` (page 3422).

Discussion
You can retrieve the window object in question by sending `object` to *notification*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSWindow.h`

windowDidBecomeMain:

Informs the delegate that the window has become main.

```
- (void)windowDidBecomeMain:(NSNotification *)notification
```

Parameters
notification
A notification named `NSWindowDidBecomeMainNotification` (page 3423).

Discussion
You can retrieve the window object in question by sending `object` to *notification*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSWindow.h

windowDidChangeScreen:

Tells the delegate that the window has changed screens.

```
- (void)windowDidChangeScreen:(NSNotification *)notification
```

Parameters
notification
A notification named NSWindowDidChangeScreenNotification (page 3423).

Discussion
You can retrieve the NSWindow object in question by sending object to *notification*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSWindow.h

windowDidChangeScreenProfile:

Tells the delegate that the window has changed screen display profiles.

```
- (void)windowDidChangeScreenProfile:(NSNotification *)notification
```

Parameters
notification
A notification named NSWindowDidChangeScreenProfileNotification (page 3423).

Discussion
You can retrieve the NSWindow object in question by sending object to *notification*.

Availability
Available in Mac OS X v10.4 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSWindow.h

windowDidDeminiaturize:

Tells the delegate that the window has been deminimized.

```
- (void)windowDidDeminiaturize:(NSNotification *)notification
```

Parameters

notification

A notification named `NSWindowDidDeminiaturizeNotification` (page 3423)

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSWindow.h`

windowDidEndLiveResize:

Informs the delegate that a live resize operation on the window has ended.

```
- (void)windowDidEndLiveResize:(NSNotification *)notification
```

Parameters

notification

A notification named `NSWindowDidEndLiveResizeNotification` (page 3424).

Discussion

You can retrieve the window object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSWindow.h`

windowDidEndSheet:

Tells the delegate that the window has closed a sheet.

```
- (void)windowDidEndSheet:(NSNotification *)notification
```

Parameters

notification

A notification named `NSWindowDidEndSheetNotification` (page 3424).

Discussion

You can retrieve the window object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.1 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSWindow.h`

windowDidExpose:

Tells the delegate that the window has been exposed.

```
- (void)windowDidExpose:(NSNotification *)notification
```

Parameters

notification

A notification named `NSWindowDidExposeNotification` (page 3424).

Discussion

You can retrieve the window object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSWindow.h`

windowDidMiniaturize:

Tells the delegate that the window has been minimized.

```
- (void)windowDidMiniaturize:(NSNotification *)notification
```

Parameters

notification

A notification named `NSWindowDidMiniaturizeNotification` (page 3425).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSWindow.h`

windowDidMove:

Tells the delegate that the window has moved.

```
- (void)windowDidMove:(NSNotification *)notification
```

Parameters

notification

A notification named `NSWindowDidMoveNotification` (page 3425).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSWindow.h

windowDidResignKey:

Informs the delegate that the window has resigned key window status.

```
- (void)windowDidResignKey:(NSNotification *)notification
```

Parameters
notification
A notification named NSWindowDidResignKeyNotification (page 3425).

Discussion
You can retrieve the window object in question by sending object to *notification*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSWindow.h

windowDidResignMain:

Informs the delegate that the window has resigned main window status.

```
- (void)windowDidResignMain:(NSNotification *)notification
```

Parameters
notification
A notification named NSWindowDidResignMainNotification (page 3425).

Discussion
You can retrieve the window object in question by sending object to *notification*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSWindow.h

windowDidResize:

Informs the delegate that the window has been resized.

```
- (void)windowDidResize:(NSNotification *)notification
```

Parameters

notification

A notification named NSWindowDidResizeNotification (page 3426).

Discussion

You can retrieve the window object in question by sending object to *notification*.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSWindow.h

windowDidUpdate:

Tells the delegate that the window received an update (page 3405) message.

```
- (void)windowDidUpdate:(NSNotification *)notification
```

Parameters

notification

A notification named NSWindowDidUpdateNotification (page 3426)

Discussion

You can retrieve the window object in question by sending object to *notification*.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

NSWindow.h

windowShouldClose:

Tells the delegate that the user has attempted to close a window or the window has received a performClose: (page 3354) message.

```
- (BOOL)windowShouldClose:(id)sender
```

Parameters

sender

The window being closed.

Return Value

YES to allow *sender* to be closed; otherwise, NO.

Discussion

This method may not always be called during window closing. Specifically, this method is not called when a user quits an application. You can find additional information on application termination in Graceful Application Termination.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSWindow.h

windowShouldZoom:toFrame:

Asks the delegate whether the specified window should zoom to the specified frame.

```
- (BOOL)windowShouldZoom:(NSWindow *)window
    toFrame:(NSRect)newFrame
```

Parameters

window
The window being zoomed.

newFrame
The rectangle to which the specified window is being zoomed.

Return Value
YES to allow *window*'s frame to become *newFrame*; otherwise, NO.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- windowWillUseStandardFrame:defaultFrame: (page 3942)

Declared In
NSWindow.h

windowWillBeginSheet:

Notifies the delegate that the window is about to open a sheet.

```
- (void)windowWillBeginSheet:(NSNotification *)notification
```

Parameters

notification
A notification named NSWindowWillBeginSheetNotification (page 3426).

Discussion
You can retrieve the window object in question by sending object to *notification*.

Availability
Available in Mac OS X v10.1 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSWindow.h

windowWillClose:

Tells the delegate that the window is about to close.

```
- (void)windowWillClose:(NSNotification *)notification
```

Parameters

notification

A notification named `NSWindowWillCloseNotification` (page 3426).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSWindow.h`

windowWillMiniaturize:

Tells the delegate that the window is about to be minimized.

```
- (void)windowWillMiniaturize:(NSNotification *)notification
```

Parameters

notification

A notification named `NSWindowWillMiniaturizeNotification` (page 3427).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.0 and later.

Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In

`NSWindow.h`

windowWillMove:

Tells the delegate that the window is about to move.

```
- (void)windowWillMove:(NSNotification *)notification
```

Parameters

notification

A notification named `NSWindowWillMoveNotification` (page 3427).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSWindow.h

windowWillResize:toSize:

Tells the delegate that the window is being resized (whether by the user or through one of the `setFrame...` methods other than `setFrame:display:` (page 3381)).

```
- (NSSize)windowWillResize:(NSWindow *)sender
    toSize:(NSSize)frameSize
```

Parameters

sender
The window being resized.

frameSize
The size to which the specified window is being resized.

Return Value
A custom size to which the specified window will be resized.

Discussion
The *frameSize* contains the size (in screen coordinates) *sender* will be resized to. To resize to a different size, simply return the desired size from this method; to avoid resizing, return the current size. *sender*'s minimum and maximum size constraints have already been applied when this method is invoked.

While the user is resizing a window, the delegate is sent a series of `windowWillResize:toSize:` messages as the window's outline is dragged. The window's outline is displayed at the constrained size as set by this method.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
NSWindow.h

windowWillReturnFieldEditor:toObject:

Tells the delegate that the field editor for a text-displaying object has been requested.

```
- (id)windowWillReturnFieldEditor:(NSWindow *)sender
    toObject:(id)client
```

Parameters

sender
The window requesting the field editor from the delegate.

client
A text-displaying object to be associated with the field editor. If `nil`, the requested field editor is the default.

Return Value
The field editor for *client*; returns `nil` when the delegate has no field editor to assign.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
`- fieldEditor:forObject:` (page 3325)

Declared In
`NSWindow.h`

windowWillReturnUndoManager:

Tells the delegate that the window's undo manager has been requested. Returns the appropriate undo manager for the window.

```
- (NSUndoManager *)windowWillReturnUndoManager:(NSWindow *)window
```

Parameters
window
The window whose undo manager is being requested.

Return Value
The appropriate undo manager for the specified window.

Discussion
If this method is not implemented by the delegate, the window creates an`NSUndoManager` for *window*.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

Declared In
`NSWindow.h`

windowWillStartLiveResize:

Informs the delegate that the window is about to be live resized.

```
- (void)windowWillStartLiveResize:(NSNotification *)notification
```

Parameters
notification
A notification named `NSWindowWillStartLiveResizeNotification` (page 3427).

Discussion
You can retrieve the window object in question by sending `object` to *notification*.

Availability
Available in Mac OS X v10.6 and later.

Declared In
NSWindow.h

windowWillUseStandardFrame:defaultFrame:

Invoked by NSWindow's zoom: (page 3409) method while determining the frame a window may be zoomed to.

```
- (NSRect)windowWillUseStandardFrame:(NSWindow *)window
    defaultFrame:(NSRect)newFrame
```

Parameters

window
The window whose frame size is being determined.

newFrame
The size of the current screen, which is the screen containing the largest part of the window's current frame, possibly reduced on the top, bottom, left, or right, depending on the current interface style. The frame is reduced on the top to leave room for the menu bar.

Return Value
The specified window's standard frame.

Discussion
The standard frame for a window should supply the size and location that are "best" for the type of information shown in the window, taking into account the available display or displays. For example, the best width for a window that displays a word-processing document is the width of a page or the width of the display, whichever is smaller. The best height can be determined similarly. On return from this method, the zoom: (page 3409) method modifies the returned standard frame, if necessary, to fit on the current screen.

Availability
Available in Mac OS X v10.0 and later.
Available as part of an informal protocol prior to Mac OS X v10.6.

See Also
- windowShouldZoom:toFrame: (page 3938)

Declared In
NSWindow.h

NSWindowScripting Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/AppKit.framework
Declared in	AppKit/NSWindowScripting.h
Companion guide	Cocoa Scripting Guide

Overview

Category on `NSWindow`. Supports window scripting for all scriptable Cocoa applications by getting and setting standard properties and handling the `close`, `print`, and `save` AppleScript commands.

Tasks

Getting Scripting Attribute Information About a Window

- `hasCloseBox` (page 3945) *required method*
  Returns `YES` if the receiver has a close box. (required)
- `hasTitleBar` (page 3946) *required method*
  Returns `YES` if the receiver has a title bar. (required)
- `isFloatingPanel` (page 3946) *required method*
  Returns `YES` if the receiver is a floating panel. (required)
- `isMiniaturizable` (page 3946) *required method*
  Returns `YES` if the receiver can be miniaturized (has a minimize button). (required)
- `isModalPanel` (page 3946) *required method*
  Returns `YES` if the receiver is an application-modal panel. (required)
- `isResizable` (page 3946) *required method*
  Returns `YES` if the receiver is resizable (has a size control). (required)
- `isZoomable` (page 3947) *required method*
  Returns `YES` if the receiver is zoomable (has a zoom button). (required)

Setting Scripting Attribute Information for a Window

- `setIsMiniaturized:` (page 3947) *required method*
  Sets the receiver's miniaturized state to the value specified by *`flag`*. (required)

- `setIsVisible:` (page 3948) *required method*
  Sets the receiver's visible state to the value specified by *`flag`*. (required)
- `setIsZoomed:` (page 3948) *required method*
  Sets the receiver's zoomed state to the value specified by *`flag`*. (required)

Handling Script Commands

- `handleCloseScriptCommand:` (page 3944) *required method*
  Handles the `close` AppleScript command by attempting to close the window (and its associated document, if any). (required)
- `handlePrintScriptCommand:` (page 3945) *required method*
  Handles the `print` AppleScript command by attempting to print the contents of the window (or its associated document, if any). (required)
- `handleSaveScriptCommand:` (page 3945) *required method*
  Handles the `save` AppleScript command by attempting to save the window (and its associated document, if any). (required)

Working with Ordered Indices

- `orderedIndex` (page 3947) *required method*
  Returns the zero-based position of the receiver based on its order from front to back among all application windows. (required)
- `setOrderedIndex:` (page 3948) *required method*
  Sets the zero-based position of the receiver, based on its order from front to back among all visible application windows, to the value specified by *`index`*. If *`index`* is out of range, sets the position to the nearest value that is in range. (required)

Instance Methods

handleCloseScriptCommand:

Handles the `close` AppleScript command by attempting to close the window (and its associated document, if any). (required)

```
- (id)handleCloseScriptCommand:(NSCloseCommand *)command
```

Discussion
Extracts `close` command arguments from the *`command`* object and uses them to determine how to close the associated document—specifically, whether to ignore unsaved changes, save changes automatically, or ask the user and to identify the file in which to save the document (by default, the file that was opened or previously saved to, or an "untitled" name if the file has never been saved).

If there is a corresponding document and the window is the main window of the document, it forwards the `close` command to the corresponding document; otherwise, the window sends itself a `performClose` message, if it has a close box. This may have been handled differently in versions of Mac OS X prior to version 10.3.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindowScripting.h`

handlePrintScriptCommand:

Handles the `print` AppleScript command by attempting to print the contents of the window (or its associated document, if any). (required)

```
- (id)handlePrintScriptCommand:(NSScriptCommand *)command
```

Discussion
If there is a corresponding document and the window is the main window of the document, it forwards the `print` command to the corresponding document; otherwise, the window sends itself a `print` message. This may have been handled differently in versions of Mac OS X prior to version 10.3.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindowScripting.h`

handleSaveScriptCommand:

Handles the `save` AppleScript command by attempting to save the window (and its associated document, if any). (required)

```
- (id)handleSaveScriptCommand:(NSScriptCommand *)command
```

Discussion
The default version invokes the same named method of the window's document, if the window is the one being saved.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindowScripting.h`

hasCloseBox

Returns `YES` if the receiver has a close box. (required)

```
- (BOOL)hasCloseBox
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindowScripting.h`

hasTitleBar

Returns `YES` if the receiver has a title bar. (required)

```
- (BOOL)hasTitleBar
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindowScripting.h`

isFloatingPanel

Returns `YES` if the receiver is a floating panel. (required)

```
- (BOOL)isFloatingPanel
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindowScripting.h`

isMiniaturizable

Returns `YES` if the receiver can be miniaturized (has a minimize button). (required)

```
- (BOOL)isMiniaturizable
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindowScripting.h`

isModalPanel

Returns `YES` if the receiver is an application-modal panel. (required)

```
- (BOOL)isModalPanel
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSWindowScripting.h`

isResizable

Returns `YES` if the receiver is resizable (has a size control). (required)

```
- (BOOL)isResizable
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindowScripting.h

isZoomable

Returns YES if the receiver is zoomable (has a zoom button). (required)

```
- (BOOL)isZoomable
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindowScripting.h

orderedIndex

Returns the zero-based position of the receiver based on its order from front to back among all application windows. (required)

```
- (int)orderedIndex
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindowScripting.h

setIsMiniaturized:

Sets the receiver's miniaturized state to the value specified by *flag*. (required)

```
- (void)setIsMiniaturized:(BOOL)flag
```

Discussion
Depending on the current miniaturized state and the value of *flag*, the window may be minimized to the Dock or expanded from the Dock.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindowScripting.h

setIsVisible:

Sets the receiver's visible state to the value specified by *flag*. (required)

```
- (void)setIsVisible:(BOOL)flag
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindowScripting.h

setIsZoomed:

Sets the receiver's zoomed state to the value specified by *flag*. (required)

```
- (void)setIsZoomed:(BOOL)flag
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindowScripting.h

setOrderedIndex:

Sets the zero-based position of the receiver, based on its order from front to back among all visible application windows, to the value specified by *index*. If *index* is out of range, sets the position to the nearest value that is in range. (required)

```
- (void)setOrderedIndex:(int)index
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSWindowScripting.h

Functions

Application Kit Functions Reference

Framework: AppKit/AppKit.h

Overview

This document describes functions and function-like macros defined in the Application Kit framework.

Functions by Task

Accessibility

Additional information on accessibility can be found in `NSAccessibility`.

`NSAccessibilityActionDescription` (page 3956)
Returns a standard description for an action.

`NSAccessibilityPostNotification` (page 3956)
Sends a notification to any observing assistive applications.

`NSAccessibilityRaiseBadArgumentException` (page 3957)
Raises an error if the parameter is the wrong type or has an illegal value

`NSAccessibilityRoleDescription` (page 3957)
Returns a standard description for a role and subrole.

`NSAccessibilityRoleDescriptionForUIElement` (page 3958)
Returns a standard role description for a user interface element.

`NSAccessibilityUnignoredChildren` (page 3959)
Returns a list of unignored accessibility objects, descending the hierarchy if necessary.

`NSAccessibilityUnignoredChildrenForOnlyChild` (page 3959)
Returns a list of unignored accessibility objects, descending the hierarchy if necessary.

`NSAccessibilityUnignoredDescendant` (page 3960)
Returns an unignored accessibility object, descending the hierarchy if necessary.

`NSAccessibilityUnignoredAncestor` (page 3958)
Returns an unignored accessibility object, ascending the hierarchy if necessary.

Applications

Additional information on `NSApplication` can be found in *NSApplication Class Reference*.

`NSApplicationLoad` (page 3960)
Startup function to call when running Cocoa code from a Carbon application.

`NSApplicationMain` (page 3961)
Called by the main function to create and run the application.

`NSPerformService` (page 3990)
Programmatically invokes a Services menu service.

`NSRegisterServicesProvider` (page 3997)
Registers a service provider.

`NSSetShowsServicesMenuItem` (page 4001)
Specifies whether an item should be included in Services menus.

`NSShowsServicesMenuItem` (page 4002)
Specifies whether a Services menu item is currently enabled.

`NSUpdateDynamicServices` (page 4003) Deprecated in Mac OS X v10.5
Causes the services information for the system to be updated.

`NSUnregisterServicesProvider` (page 4002) Deprecated in Mac OS X v10.6
Unregisters a service provider.

Events

`NSEventMaskFromType` (page 3981)
Returns the event mask for the specified type.

Fonts

`NSConvertGlyphsToPackedGlyphs` (page 3967)
Prepares a set of glyphs for processing by character-based routines.

Graphics

`NSCopyBits` (page 3967)
Copies a bitmap image to the location specified by a destination point.

`NSCountWindows` (page 3968)
Counts the number of onscreen windows.

`NSCountWindowsForContext` (page 3968)
Counts the number of onscreen windows belonging to a particular application.

`NSDisableScreenUpdates` (page 3969)
Disables screen updates.

`NSEnableScreenUpdates` (page 3980)
Enables screen updates

`NSDottedFrameRect` (page 3970)
Draws a bordered rectangle.

`NSDrawBitmap` (page 3970)
Draws a bitmap image.

NSDrawButton (page 3972)
Draws a gray-filled rectangle representing a user-interface button.

NSDrawDarkBezel (page 3973)
Draws a dark gray-filled rectangle with a bezel border.

NSDrawGrayBezel (page 3973)
Draws a gray-filled rectangle with a bezel border.

NSDrawGroove (page 3974)
Draws a gray-filled rectangle with a groove border.

NSDrawLightBezel (page 3974)
Draws a white-filled rectangle with a bezel border.

NSDrawThreePartImage (page 3976)
Draws a three-part tiled image.

NSDrawNinePartImage (page 3975)
Draws a nine-part tiled image.

NSDrawTiledRects (page 3978)
Draws rectangles with borders.

NSDrawColorTiledRects (page 3972)
Draws a colored bordered rectangle.

NSDrawWhiteBezel (page 3979)
Draws a white-filled rectangle with a bezel border.

NSDrawWindowBackground (page 3980)
Draws the window's default background pattern into the specified rectangle of the currently focused view.

NSEraseRect (page 3980)
Erases the specified rect by filling it with white.

NSFrameRectWithWidth (page 3982)
Draw a bordered rectangle.

NSFrameRectWithWidthUsingOperation (page 3983)
Draw a bordered rectangle using the specified compositing operation.

NSHighlightRect (page 3987)
Highlights the specified rect by filling it with white.

NSReadPixel (page 3991)
Reads the color of the pixel at the specified location.

NSRectClip (page 3991)
Modifies the current clipping path by intersecting it with the passed rect.

NSRectClipList (page 3992)
Modifies the current clipping path by intersecting it with the passed rect.

NSRectFill (page 3993)
Fills the passed rectangle with the current color.

NSRectFillList (page 3993)
Fills the rectangles in the passed list with the current fill color.

NSRectFillListWithColors (page 3994)
Fills the rectangles in the passed list with the passed list of colors.

`NSRectFillListWithGrays` (page 3996)
Fills the rectangles in the passed list with the passed list of grays.

`NSRectFillListUsingOperation` (page 3994)
Fills the rectangles in a list using the current fill color and specified compositing operation.

`NSRectFillListWithColorsUsingOperation` (page 3995)
Fills the rectangles in a list using the specified colors and compositing operation.

`NSRectFillUsingOperation` (page 3997)
Fills a rectangle using the current fill color and the specified compositing operation.

`NSSetFocusRingStyle` (page 4000)
Specifies how a focus ring will be drawn.

`NSShowAnimationEffect` (page 4001)
Runs a system animation effect.

`NSWindowList` (page 4003)
Gets information about onscreen windows.

`NSWindowListForContext` (page 4003)
Gets information about an application's onscreen windows.

`NSFrameRect` (page 3981) Available in Mac OS X v10.3 through Mac OS X v10.5
Draw a bordered rectangle.

`NSGetWindowServerMemory` (page 3986) Deprecated in Mac OS X v10.6
Returns the amount of memory being used by a context.

Graphics-Window Depth

`NSAvailableWindowDepths` (page 3962)
Returns the available window depth values.

`NSBestDepth` (page 3965)
Attempts to return a window depth adequate for the specified parameters.

`NSBitsPerPixelFromDepth` (page 3966)
Returns the bits per pixel for the specified window depth.

`NSBitsPerSampleFromDepth` (page 3966)
Returns the bits per sample for the specified window depth.

`NSColorSpaceFromDepth` (page 3966)
Returns the name of the color space corresponding to the passed window depth.

`NSNumberOfColorComponents` (page 3988)
Returns the number of color components in the specified color space.

`NSPlanarFromDepth` (page 3990)
Returns whether the specified window depth is planar.

Interface Styles

`NSInterfaceStyleForKey` (page 3987)
Returns an interface style value for the specified key and responder.

Key Value Bindings

`NSIsControllerMarker` (page 3988)
Tests whether a given object is special marker object used for indicating the state of a selection in relation to a key.

OpenGL

`NSOpenGLGetOption` (page 3989)
Returns global OpenGL options.

`NSOpenGLGetVersion` (page 3989)
Returns the NSOpenGL version numbers.

`NSOpenGLSetOption` (page 3989)
Sets global OpenGL options.

Panels

`NSBeginAlertSheet` (page 3962)
Creates and runs an alert sheet.

`NSBeginCriticalAlertSheet` (page 3964)
Creates and runs a critical alert sheet.

`NSBeginInformationalAlertSheet` (page 3964)
Creates and runs an informational alert sheet.

`NSGetAlertPanel` (page 3983)
Returns an alert panel.

`NSGetCriticalAlertPanel` (page 3984)
Returns an alert panel to display a critical message.

`NSGetInformationalAlertPanel` (page 3985)
Returns an alert panel to display an informational message.

`NSReleaseAlertPanel` (page 3998)
Releases an alert panel.

`NSRunAlertPanel` (page 3998)
Creates an alert panel.

`NSRunCriticalAlertPanel` (page 3999)
Creates and runs a critical alert panel.

`NSRunInformationalAlertPanel` (page 4000) Deprecated in Mac OS X v10.4
Creates and runs an informational alert panel.

Pasteboards

`NSCreateFileContentsPboardType` (page 3968)
Returns a pasteboard type based on the passed file type. (Deprecated. The file contents pboard type allowed you to synthesize a pboard type for a file's contents based on the file's extension. Using the UTI of a file to represent its contents now replaces this functionality.)

`NSCreateFilenamePboardType` (page 3969)
Returns a pasteboard type based on the passed file type. (Deprecated. The file contents pboard type allowed you to synthesize a pboard type for a file's contents based on the file's extension. Using the UTI of a file to represent its contents now replaces this functionality.)

`NSGetFileType` (page 3985)
Returns a file type based on the passed pasteboard type. (Deprecated. The file contents pboard type allowed you to synthesize a pboard type for a file's contents based on the file's extension. Using the UTI of a file to represent its contents now replaces this functionality.)

`NSGetFileTypes` (page 3985)
Returns an array of file types based on the passed pasteboard types. (Deprecated. The file contents pboard type allowed you to synthesize a pboard type for a file's contents based on the file's extension. Using the UTI of a file to represent its contents now replaces this functionality.)

System Beep

Additional information on sounds can be found in `NSSound`.

`NSBeep` (page 3962)
Plays the system beep.

Functions

NSAccessibilityActionDescription

Returns a standard description for an action.

```
NSString * NSAccessibilityActionDescription (
   NSString *action
);
```

Discussion
This function returns a standard description for *action*.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
Dicey
ImageMap
ImageMapExample

Declared In
`NSAccessibility.h`

NSAccessibilityPostNotification

Sends a notification to any observing assistive applications.

```
void NSAccessibilityPostNotification (
   id element,
   NSString *notification
);
```

Discussion
Sends *notification* to any assistive applications that have registered to receive the notification from the user interface object *element* in your application. Accessibility notifications require special handling, so they cannot be posted using `NSNotificationCenter`.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
Dicey
Sketch+Accessibility
TrackBall

Declared In
`NSAccessibility.h`

NSAccessibilityRaiseBadArgumentException

Raises an error if the parameter is the wrong type or has an illegal value

```
void NSAccessibilityRaiseBadArgumentException (
   id element,
   NSString *attribute,
   id value
);
```

Discussion
Raises an error if a parameter is the wrong type or has an illegal value. This function can also be used to raise an error if an attempt is made to set an attribute's value with the wrong type or an illegal value.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
Sketch+Accessibility

Declared In
`NSAccessibility.h`

NSAccessibilityRoleDescription

Returns a standard description for a role and subrole.

```
NSString * NSAccessibilityRoleDescription (
   NSString *role,
   NSString *subrole
);
```

Discussion
You should pass nil to this function if there is no subrole. This function returns a description of a standard role. For example, if you implement a button widget that does not inherit from `NSButton`, you should use this function to return a localized role description matching that returned by a standard button.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
Dicey
ImageMap
ImageMapExample
Sketch+Accessibility
ZipBrowser

Declared In
`NSAccessibility.h`

NSAccessibilityRoleDescriptionForUIElement

Returns a standard role description for a user interface element.

```
NSString * NSAccessibilityRoleDescriptionForUIElement (
   id element
);
```

Discussion
This function is like the `NSAccessibilityRoleDescription` (page 3957) function, except that it queries *`element`* to get the role and subrole. The `NSAccessibilityRoleDescription` function is more efficient, but this function is useful for accessorizing base classes so that they properly handle derived classes, which may override the subrole or even the role.

Availability
Available in Mac OS X v10.4 and later.

Related Sample Code
ImageMap
ImageMapExample

Declared In
`NSAccessibility.h`

NSAccessibilityUnignoredAncestor

Returns an unignored accessibility object, ascending the hierarchy if necessary.

```
id NSAccessibilityUnignoredAncestor (
    id element
);
```

Discussion
Tests whether *element* is an ignored object, returning either *element*, if it is not ignored, or the first unignored ancestor of *element*.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
ImageMap
ImageMapExample
Sketch+Accessibility
TrackBall
ZipBrowser

Declared In
NSAccessibility.h

NSAccessibilityUnignoredChildren

Returns a list of unignored accessibility objects, descending the hierarchy if necessary.

```
NSArray * NSAccessibilityUnignoredChildren (
   NSArray *originalChildren
);
```

Discussion
Returns a copy of *originalChildren* with any ignored objects in the array replaced by their unignored descendants.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
Dicey
ImageMap
ImageMapExample

Declared In
NSAccessibility.h

NSAccessibilityUnignoredChildrenForOnlyChild

Returns a list of unignored accessibility objects, descending the hierarchy if necessary.

```
NSArray * NSAccessibilityUnignoredChildrenForOnlyChild (
   id originalChild
);
```

Discussion
Tests whether *originalChild* is an ignored object and returns an array containing either *originalChild*, if it is not ignored, or its unignored descendants.

Availability
Available in Mac OS X v10.2 and later.

Declared In
NSAccessibility.h

NSAccessibilityUnignoredDescendant

Returns an unignored accessibility object, descending the hierarchy if necessary.

```
id NSAccessibilityUnignoredDescendant (
   id element
);
```

Discussion
Tests whether *element* is an ignored object, returning either *element*, if it is not ignored, or the first unignored descendant of *element*. Use this function only if you know there is a linear, one-to-one, hierarchy below *element*. Otherwise, if *element* has either no unignored children or multiple unignored children, this function fails and returns nil.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
ImageMap
ImageMapExample

Declared In
NSAccessibility.h

NSApplicationLoad

Startup function to call when running Cocoa code from a Carbon application.

```
BOOL NSApplicationLoad (void);
```

Return Value
YES if the NSApplication object was successfully initialized and can now be used from your Carbon application or NO if there was an error during initialization.

Discussion
You typically call this function before calling other Cocoa code in a plug-in loaded into a primarily Carbon application. If the shared NSApplication object is not already initialized, this function initializes it and sets up the necessary event handlers for Cocoa.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
SpellingChecker-CocoaCarbon

Declared In
`NSApplication.h`

NSApplicationMain

Called by the main function to create and run the application.

```
int NSApplicationMain (
   int argc,
   const char *argv[]
);
```

Parameters

argc

The number of arguments in the argv parameter.

argv

An array of pointers containing the arguments that were passed to the application at startup.

Return Value
This method never returns a result code. Instead, it calls the `exit` function to exit the application and terminate the process. If you want to determine why the application exited, you should look at the result code from the `exit` function instead.

Discussion
Creates the application, loads the main nib file from the application's main bundle, and runs the application. You must call this function from the main thread of your application, and you typically call it only once from your application's `main` function, which is usually generated automatically by Xcode.

Special Considerations

`NSApplicationMain` itself ignores the *argc* and *argv* arguments. Instead, Cocoa gets its arguments indirectly via `_NSGetArgv`, `_NSGetArgc`, and `_NSGetEnviron` (see <crt_externs.h>).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Cocoa Tips and Tricks
CoreRecipes
ImageClient
ImageKitDemo
MyPhoto

Declared In
`NSApplication.h`

NSAvailableWindowDepths

Returns the available window depth values.

```
const NSWindowDepth * NSAvailableWindowDepths (void);
```

Discussion
Returns a null-terminated array of `NSWindowDepthâ  Window Depth` (page 3416) values that specify which window depths are currently available. Window depth values are converted to specific display properties using the functions `NSBitsPerPixelFromDepth` (page 3966), `NSBitsPerSampleFromDepth` (page 3966), `NSColorSpaceFromDepth` (page 3966), and `NSPlanarFromDepth` (page 3990).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphics.h`

NSBeep

Plays the system beep.

```
void NSBeep (void);
```

Discussion
Plays the system beep. Users can select a sound to be played as the system beep. On a Macintosh computer, for example, you can change sounds with the Sound pane of System Preferences.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CoreRecipes
GridCalendar
NewsReader
PDFKitLinker2
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSGraphics.h`

NSBeginAlertSheet

Creates and runs an alert sheet.

```
void NSBeginAlertSheet (
   NSString *title,
   NSString *defaultButton,
   NSString *alternateButton,
   NSString *otherButton,
   NSWindow *docWindow,
   id modalDelegate,
   SEL didEndSelector,
   SEL didDismissSelector,
   void *contextInfo,
   NSString *msg,
   ...
);
```

Discussion
Creates and runs an alert sheet on *docWindow*, with the title of *title*, the text of *msg*, and buttons with titles of *defaultButton*, *alternateButton*, and *otherButton*.

The buttons are laid out on the lower-right corner of the sheet, with *defaultButton* on the right, *alternateButton* on the left, and *otherButton* in the middle. If *title* is `nil` or an empty string, a default localized title is used ("Alert" in English). If *defaultButton* is `nil` or an empty string, a default localized button title ("OK" in English) is used. For the remaining buttons, this function creates them only if their corresponding button title is non-`nil`.

A Command-D key equivalent for the "Don't Save" button is provided, if one is found. The button titles are searched for the localized value for "Don't Save." If a match is found, that button is assigned a Command-D key equivalent, provided it is not the default button.

If you create a modal panel using `runModalForWindow:` (page 163) or `beginSheet:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 140), you can assign the key equivalent yourself, using `setKeyEquivalent:` (page 483) and `setKeyEquivalentModifierMask:` (page 484).

The *msg* argument is the message that's displayed in the panel. It can use printf-style formatting characters; any necessary arguments should be listed at the end of the function's argument list (after the *msg* argument). For more information on formatting characters, see the man page for `printf`.

When the modal session is ended, and before the sheet is dismissed, the *didEndSelector* is invoked on the *modalDelegate*. passing *contextInfo*. After the sheet is dismissed, the *didDismissSelector* is invoked on the *modalDelegate*, passing *contextInfo*. Typically, you will want to implement the *didEndSelector* but you may pass `NULL` for the *didDismissSelector*. The two selectors should be defined as follows:

```
sheetDidEnd:(NSWindow *)sheet returnCode:(int)returnCode contextInfo:(void
*)contextInfo;
sheetDidDismiss:(NSWindow *)sheet returnCode:(int)returnCode contextInfo:(void
  *)contextInfo;
```

where *sheet* is the alert sheet, *returnCode* specifies which button the user pressed, and *contextInfo* is the same *contextInfo* passed into `NSBeginAlertSheet`. *returnCode* can be one of the following:

- `NSAlertDefaultReturn` means the user pressed the default button.
- `NSAlertAlternateReturn` means the user pressed the alternate button.
- `NSAlertOtherReturn` means the user pressed the other button.

- `NSAlertErrorReturn` means an error occurred while running the alert panel.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ImageClient
PrefsPane
QTAudioExtractionPanel
QTKitPlayer
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSPanel.h`

NSBeginCriticalAlertSheet

Creates and runs a critical alert sheet.

```
void NSBeginCriticalAlertSheet (
   NSString *title,
   NSString *defaultButton,
   NSString *alternateButton,
   NSString *otherButton,
   NSWindow *docWindow,
   id modalDelegate,
   SEL didEndSelector,
   SEL didDismissSelector,
   void *contextInfo,
   NSString *msg,
   ...
);
```

Discussion
Creates and runs a critical alert sheet on *docWindow*, with the title of *title*, the text of *msg*, and buttons with titles of *defaultButton*, *alternateButton*, and *otherButton*.

See the description of `NSBeginAlertSheet` (page 3962) for information on layout, default parameters, and the selectors.

The sheet presented to the user is badged with a caution icon. Critical alerts should be used only as specified in the "Alerts" section of the Windows chapter of *Apple Human Interface Guidelines*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPanel.h`

NSBeginInformationalAlertSheet

Creates and runs an informational alert sheet.

```
void NSBeginInformationalAlertSheet (
   NSString *title,
   NSString *defaultButton,
   NSString *alternateButton,
   NSString *otherButton,
   NSWindow *docWindow,
   id modalDelegate,
   SEL didEndSelector,
   SEL didDismissSelector,
   void *contextInfo,
   NSString *msg,
   ...
);
```

Discussion
Creates and runs an informational alert sheet on *docWindow*, with the title of *title*, the text of *msg*, and buttons with titles of *defaultButton*, *alternateButton*, and *otherButton*.

See the description of `NSBeginAlertSheet` (page 3962) for information on layout, default parameters, and the selectors.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AutoSample
InstallerPluginSample
SimpleToolbar

Declared In
`NSPanel.h`

NSBestDepth

Attempts to return a window depth adequate for the specified parameters.

```
NSWindowDepth NSBestDepth (
   NSString *colorSpace,
   NSInteger bps,
   NSInteger bpp,
   BOOL planar,
   BOOL *exactMatch
);
```

Discussion
Returns a window depth deep enough for the given number of colors in *colorSpace*, bits per sample specified by *bps*, bits per pixel specified by *bpp*, and whether planar as specified by *planar*. Upon return, the variable pointed to by *exactMatch* is `YES` if the window depth can accommodate all of the values specified by the parameters, `NO` if it can't.

Use this function to compute window depths. This function tries to accommodate all the parameters (match or better); if there are multiple matches, it gives the closest, with matching *colorSpace* first, then *bps*, then *planar*, then *bpp*. *bpp* is "bits per pixel"; 0 indicates default (same as the number of bits per plane, either

bps or bps * NSNumberOfColorComponents (page 3988)); other values may be used as hints to provide backing stores of different configuration; for instance, 8-bit color. The *exactMatch* parameter is optional and indicates whether all the parameters matched exactly.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSGraphics.h

NSBitsPerPixelFromDepth

Returns the bits per pixel for the specified window depth.

```
NSInteger NSBitsPerPixelFromDepth (
   NSWindowDepth depth
);
```

Discussion
Returns the number of bits per pixel for the window depth specified by *depth*.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
GLUT

Declared In
NSGraphics.h

NSBitsPerSampleFromDepth

Returns the bits per sample for the specified window depth.

```
NSInteger NSBitsPerSampleFromDepth (
   NSWindowDepth depth
);
```

Discussion
Returns the number of bits per sample (bits per pixel in each color component) for the window depth specified by *depth*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSGraphics.h

NSColorSpaceFromDepth

Returns the name of the color space corresponding to the passed window depth.

```
NSString * NSColorSpaceFromDepth (
   NSWindowDepth depth
);
```

Discussion
Returns the color space name for the specified *depth*. For example, the returned color space name can be `NSCalibratedRGBColorSpace`, or `NSDeviceCMYKColorSpace`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphics.h`

NSConvertGlyphsToPackedGlyphs

Prepares a set of glyphs for processing by character-based routines.

```
NSInteger NSConvertGlyphsToPackedGlyphs (
   NSGlyph *glBuf,
   NSInteger count,
   NSMultibyteGlyphPacking packing,
   char *packedGlyphs
);
```

Discussion
Takes a buffer of glyphs, specified in the *glBuf* parameter, and packs them into a condensed character array. The character array is returned in the *packedGlyphs* parameter, which should have enough space for at least ((4 * count) + 1) bytes to guarantee that the packed glyphs fit. *count* specifies the number of glyphs in *glBuf*. *packing* specifies how the glyphs are currently packed.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSFont.h`

NSCopyBits

Copies a bitmap image to the location specified by a destination point.

```
void NSCopyBits (
   NSInteger srcGState,
   NSRect srcRect,
   NSPoint destPoint
);
```

Discussion
Copies the pixels in the rectangle specified by *srcRect* to the location specified by *destPoint*. The source rectangle is defined in the graphics state designated by *srcGState*. If *srcGState* is `NSNullObject`, the current graphics state is assumed. The *destPoint* destination is defined in the current graphics state.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSGraphics.h

NSCountWindows

Counts the number of onscreen windows.

```
void NSCountWindows (
   NSInteger *count
);
```

Parameters

count

On output, this parameter contains the number of onscreen windows.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSGraphics.h

NSCountWindowsForContext

Counts the number of onscreen windows belonging to a particular application.

```
void NSCountWindowsForContext (
   NSInteger context,
   NSInteger *count
);
```

Discussion
Counts the number of onscreen windows belonging to a particular application, identified by *context*, which is a window server connection ID. The function returns the number by reference in *count*.

Use of this function is discouraged as it may be deprecated in a future release.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSGraphics.h

NSCreateFileContentsPboardType

Returns a pasteboard type based on the passed file type. (Deprecated. The file contents pboard type allowed you to synthesize a pboard type for a file's contents based on the file's extension. Using the UTI of a file to represent its contents now replaces this functionality.)

```
NSString * NSCreateFileContentsPboardType (
   NSString *fileType
);
```

Discussion
Returns an NSString to a pasteboard type representing a file's contents based on the supplied string *fileType*. *fileType* should generally be the extension part of a filename. The conversion from a named file type to a pasteboard type is simple; no mapping to standard pasteboard types is attempted.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPasteboard.h

NSCreateFilenamePboardType

Returns a pasteboard type based on the passed file type. (Deprecated. The file contents pboard type allowed you to synthesize a pboard type for a file's contents based on the file's extension. Using the UTI of a file to represent its contents now replaces this functionality.)

```
NSString * NSCreateFilenamePboardType (
   NSString *fileType
);
```

Discussion
Returns an NSString to a pasteboard type representing a filename based on the supplied string *fileType*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPasteboard.h

NSDisableScreenUpdates

Disables screen updates.

```
void NSDisableScreenUpdates (void);
```

Discussion
Prevents drawing operations from being flushed to the screen for all windows belonging to the calling process. When you re-enable screen updates (with NSEnableScreenUpdates (page 3980)) screen flushing for all windows belonging to the calling process appears to be simultaneous. You typically call this function so that operations on multiple windows appear atomic to the user. This is a technique particularly useful for synchronizing parent and child windows. Make sure that the period after calling this function and before reenabling updates is short; the system only allow updating to be disabled for a limited time (currently one second) before automatically reenabling updates. Successive calls to this function are placed on a stack and must be popped off that stack by matching NSEnableScreenUpdates calls.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
AnimatedTableView

Declared In
NSGraphics.h

NSDottedFrameRect

Draws a bordered rectangle.

```
void NSDottedFrameRect (
   NSRect aRect
);
```

Discussion
Deprecated. Use a dashed `NSBezierPath` instead.

Availability
Available in Mac OS X v10.0 and later.

See Also
`NSDrawTiledRects` (page 3978)

Declared In
NSGraphics.h

NSDrawBitmap

Draws a bitmap image.

```
void NSDrawBitmap (
   NSRect rect,
   NSInteger width,
   NSInteger height,
   NSInteger bps,
   NSInteger spp,
   NSInteger bpp,
   NSInteger bpr,
   BOOL isPlanar,
   BOOL hasAlpha,
   NSString *colorSpaceName,
   const unsigned char *const data[5]
);
```

Discussion
This function is marginally obsolete. Most applications are better served using the `NSBitmapImageRep` class to read and display bitmap images.

This function renders an image from a bitmap, binary data that describes the pixel values for the image.

This function renders a bitmap image using an appropriate display operator. It puts the image in the rectangular area specified by its first argument, *rect*; the rectangle is specified in the current coordinate system and is located in the current window. The next two arguments, *pixelsWide* and *pixelsHigh*, give the width and height of the image in pixels. If either of these dimensions is larger or smaller than the corresponding dimension of the destination rectangle, the image will be scaled to fit.

The remaining arguments describe the bitmap data, as explained in the following paragraphs.

The *bitsPerSample* argument is the number of bits per sample for each pixel and *samplesPerPixel* is the number of samples per pixel. *bitsPerPixel* is based on *samplesPerPixel* and the configuration of the bitmap: if the configuration is planar, then the value of *bitsPerPixel* should equal the value of *bitsPerSample*; if the configuration isn't planar (is meshed instead), *bitsPerPixel* should equal `bitsPerSample * samplesPerPixel`.

The *bytesPerRow* argument is calculated in one of two ways, depending on the configuration of the image data (data configuration is described below). If the data is planar, `bytesPerRow is (7 + (pixelsWide * bitsPerSample)) / 8`. If the data is meshed, `bytesPerRow is (7 + (pixelsWide * bitsPerSample * samplesPerPixel)) / 8`.

A sample is data that describes one component of a pixel. In an RGB color system, the red, green, and blue components of a color are specified as separate samples, as are the cyan, magenta, yellow, and black components in a CMYK system. Color values in a grayscale are a single sample. Alpha values that determine transparency and opaqueness are specified as a coverage sample separate from color. In bitmap images with alpha, the color (or gray) components have to be premultiplied with the alpha. This is the way images with alpha are displayed, this is the way they are read back, and this is the way they are stored in TIFFs.

The *isPlanar* argument refers to the way data is configured in the bitmap. This flag should be set to `YES` if a separate data channel is used for each sample. The function provides for up to five channels, *data1*, *data2*, *data3*, *data4*, and *data5*. It should be set `NO` if sample values are interwoven in a single channel (meshed); all values for one pixel are specified before values for the next pixel.

Grayscale windows store pixel data in planar configuration; color windows store it in meshed configuration. `NSDrawBitmap` can render meshed data in a planar window, or planar data in a meshed window. However, it's more efficient if the image has a depth (*bitsPerSample*) and configuration (*isPlanar*) that match the window.

The *hasAlpha* argument indicates whether the image contains alpha. If it does, the number of samples should be 1 greater than the number of color components in the model (for example, 4 for RGB).

The *colorSpace* argument can be `NSCustomColorSpace`, indicating that the image data is to be interpreted according to the current color space in the graphics state. This allows for imaging using custom color spaces. The image parameters supplied as the other arguments should match what the color space is expecting.

If the image data is planar, *data*[0] through *data*[*samplesPerPixel*–1] point to the planes; if the *data* is meshed, only *data*[0] needs to be set.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphics.h`

NSDrawButton

Draws a gray-filled rectangle representing a user-interface button.

```
void NSDrawButton (
   NSRect aRect,
   NSRect clipRect
);
```

Parameters

aRect

The bounding rectangle (in the current coordinate system) in which to draw. Only those parts of *aRect* that lie within the *clipRect* are actually drawn.

clipRect

The clipping rectangle to use during drawing.

Discussion

Draws a gray-filled rectangle, used to signify a user-interface button. Since this function is often used to draw the border of a view, the *aRect* parameter typically contains the view's bounds rectangle. For an Aqua button, use an `NSButton` object instead.

This function fills the specified rectangle with light gray. This function is designed for rectangles that are defined in unscaled, unrotated coordinate systems (that is, where the y axis is vertical, the x axis is horizontal, and a unit along either axis is equal to 1 screen pixel). The coordinate system can be either flipped or unflipped. The sides of the rectangle should lie on pixel boundaries.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSGraphics.h`

NSDrawColorTiledRects

Draws a colored bordered rectangle.

```
NSRect NSDrawColorTiledRects (
   NSRect boundsRect,
   NSRect clipRect,
   const NSRectEdge *sides,
   NSColor **colors,
   NSInteger count
);
```

Parameters

boundsRect

The bounding rectangle (in the current coordinate system) in which to draw. Since this function is often used to draw the border of a view, this rectangle will typically be that view's bounds rectangle. Only those parts of *boundsRect* that lie within the *clipRect* are actually drawn.

clipRect

The clipping rectangle to use during drawing.

sides

The sides of the rectangle for which you want to specify custom colors. Each side must have a corresponding entry in the *colors* parameter.

colors
The colors to draw for each of the edges listed in the *sides* parameter.

count
The number of 1.0-unit-wide slices to draw on the specified sides.

Return Value
The rectangle that lies within the resulting border.

Discussion
Behaves the same as `NSDrawTiledRects` (page 3978) except it draws its border using colors from the *colors* array.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphics.h`

NSDrawDarkBezel

Draws a dark gray-filled rectangle with a bezel border.

```
void NSDrawDarkBezel (
   NSRect aRect,
   NSRect clipRect
);
```

Parameters

aRect
The bounding rectangle (in the current coordinate system) in which to draw. Only those parts of *aRect* that lie within the *clipRect* are actually drawn.

clipRect
The clipping rectangle to use during drawing.

Availability
Available in Mac OS X v10.0 and later.

See Also
`NSDrawTiledRects` (page 3978)

Related Sample Code
BindingsJoystick

Declared In
`NSGraphics.h`

NSDrawGrayBezel

Draws a gray-filled rectangle with a bezel border.

```
void NSDrawGrayBezel (
   NSRect aRect,
   NSRect clipRect
);
```

Parameters

aRect

The bounding rectangle (in the current coordinate system) in which to draw. Only those parts of *aRect* that lie within the *clipRect* are actually drawn.

clipRect

The clipping rectangle to use during drawing.

Availability
Available in Mac OS X v10.0 and later.

See Also
NSDrawTiledRects (page 3978)

Declared In
NSGraphics.h

NSDrawGroove

Draws a gray-filled rectangle with a groove border.

```
void NSDrawGroove (
   NSRect aRect,
   NSRect clipRect
);
```

Parameters

aRect

The bounding rectangle (in the current coordinate system) in which to draw. Only those parts of *aRect* that lie within the *clipRect* are actually drawn.

clipRect

The clipping rectangle to use during drawing.

Availability
Available in Mac OS X v10.0 and later.

See Also
NSDrawTiledRects (page 3978)

Declared In
NSGraphics.h

NSDrawLightBezel

Draws a white-filled rectangle with a bezel border.

```
void NSDrawLightBezel (
   NSRect aRect,
   NSRect clipRect
);
```

Parameters

aRect

The bounding rectangle (in the current coordinate system) in which to draw. Only those parts of *aRect* that lie within the *clipRect* are actually drawn.

clipRect

The clipping rectangle to use during drawing.

Availability
Available in Mac OS X v10.0 and later.

See Also
NSDrawTiledRects (page 3978)

Related Sample Code
BindingsJoystick

Declared In
NSGraphics.h

NSDrawNinePartImage

Draws a nine-part tiled image.

```
void NSDrawNinePartImage(NSRect frame,
   NSImage *topLeftCorner,
   NSImage *topEdgeFill,
   NSImage *topRightCorner,
   NSImage *leftEdgeFill,
   NSImage *centerFill,
   NSImage *rightEdgeFill,
   NSImage *bottomLeftCorner,
   NSImage *bottomEdgeFill,
   NSImage *bottomRightCorner,
   NSCompositingOperation op,
   CGFloat alphaFraction,
   BOOL flipped
);
```

Parameters

frame

The rectangle (specified in the current coordinate system) in which to draw the images.

topLeftCorner

The image to display in the top-left corner.

topEdgeFill

The image used to tile the space between the *topLeftCorner* and *topRightCorner* images.

topRightCorner

The image to display in the top-right corner.

leftEdgeFill
The image used to tile the space between the *topLeftCorner* and *bottomLeftCorner* images.

centerFill
The image used to tile the center area between the other eight images.

rightEdgeFill
The image used to tile the space between the *topRightCorner* and *bottomRightCorner* images.

bottomLeftCorner
The image to display in the bottom-left corner.

bottomEdgeFill
The image used to tile the space between the *bottomLeftCorner* and *bottomRightCorner* images.

bottomRightCorner
The image to display in the bottom-right corner.

op
The compositing operation to use when rendering the images.

alphaFraction
The alpha value to apply to the rendered image. This value can range between 0.0 and 1.0, with 0.0 being fully transparent and 1.0 being fully opaque.

flipped
Specify `YES` if you are drawing the images in a flipped coordinate system; otherwise, specify `NO`.

Discussion
This function is typically used to draw custom cells that are capable of being resized both vertically and horizontally. Cells of this type are comprised of four fixed-size corner images along and a set of edge and center images that are used to fill the gaps between the corners. These cells allow you to create sophisticated looking controls that can grow and shrink in any direction without distorting the control's overall appearance.

You should prefer the use of this function over your own custom code for handling multi-part images whose size can change. This function correctly manages the subtle behaviors needed to handle resolution independence issues and to avoid visual artifacts caused by tiling the various images.

This function uses the top-left and bottom-right corner images to determine the widths and heights of the edge areas that need to be filled. If the width or height of the bottom-left and top-right images are not sized appropriately, they may be scaled to fill their corner area. Edge areas between the corners are tiled using the corresponding image. Similarly, the center area is tiled using the specified center image.

The *flipped* parameter lets you reorient the contents of each image when drawing in a flipped coordinate system. By default, images use an internal coordinate system that is not flipped. Rendering such an image in a flipped coordinate system would therefore cause the image to appear upside down. Passing `YES` for the *flipped* parameter adjusts the image's internal coordinate system to draw it correctly in a flipped environment.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSCell.h`

NSDrawThreePartImage

Draws a three-part tiled image.

```
void NSDrawThreePartImage(NSRect frame,
   NSImage *startCap,
   NSImage *centerFill,
   NSImage *endCap,
   BOOL vertical,
   NSCompositingOperation op,
   CGFloat alphaFraction,
   BOOL flipped
);
```

Parameters

frame

The rectangle (specified in the current coordinate system) in which to draw the images.

startCap

For a horizontal three-part image, this is the image located at the left edge of the frame rectangle. For a vertical three-part image, this image appears at the top of the screen in an unflipped coordinate system and at the bottom of the screen in a flipped coordinate system.

centerFill

The image used to tile the space between the *startCap* and *endCap* images.

endCap

For a horizontal three-part image, this is the image located at the right edge of the frame rectangle. For a vertical three-part image, this image appears at the bottom of the screen in an unflipped coordinate system and at the top of the screen in a flipped coordinate system.

vertical

Specify `YES` if the images should be stacked on top of one another to create a vertically oriented element. Specify `NO` if the images should be laid out side-by-side to create a horizontally oriented element.

op

The compositing operation to use when rendering the images.

alphaFraction

The alpha value to apply to the rendered image. This value can range between 0.0 and 1.0, with 0.0 being fully transparent and 1.0 being fully opaque.

flipped

Specify `YES` if you are drawing the images in a flipped coordinate system; otherwise, specify `NO`.

Discussion

This function is typically used to draw custom cells (such as the backgrounds for push button and slider controls) that are capable of being resized along a single axis only. Cells of this type are comprised of fixed-size end cap images and a center area that is filled by tiling the specified center image as many times as needed to fill the gap. These cells allow you to create sophisticated looking controls that can grow and shrink without distorting the control's overall appearance.

You should prefer the use of this function over your own custom code for handling multi-part images whose size can change. This function correctly manages the subtle behaviors needed to handle resolution independence issues and to avoid visual artifacts caused by tiling the various images.

When drawing a horizontally oriented control, the images in the *startCap*, *centerFill*, and *endCap* parameters should all have the same height, and that height should match the height of the frame rectangle. If an image's height does not match the height of the frame rectangle, it is scaled until it does match, which might yield less desirable results. For vertically oriented controls, the image widths are scaled instead of the heights.

The *flipped* parameter lets you reorient the contents of each image when drawing in a flipped coordinate system. By default, images use an internal coordinate system that is not flipped. Rendering such an image in a flipped coordinate system would therefore cause the image to appear upside down. Passing YES for the *flipped* parameter adjusts the image's internal coordinate system to draw it correctly in a flipped environment.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSCell.h

NSDrawTiledRects

Draws rectangles with borders.

```
NSRect NSDrawTiledRects (
   NSRect boundsRect,
   NSRect clipRect,
   const NSRectEdge *sides,
   const CGFloat *grays,
   NSInteger count
);
```

Parameters

boundsRect
The bounding rectangle (in the current coordinate system) in which to draw. Since this function is often used to draw the border of a view, this rectangle will typically be that view's bounds rectangle. Only those parts of *boundsRect* that lie within the *clipRect* are actually drawn.

clipRect
The clipping rectangle to use during drawing.

sides
The sides of the rectangle for which you want to specify custom gray levels. Each side must have a corresponding entry in the *grays* parameter.

grays
The gray levels to draw for each of the edges listed in the *sides* parameter.

count
The number of 1.0-unit-wide slices to draw on the specified sides.

Return Value
The rectangle that lies within the resulting border.

Discussion
This is a generic function that can be used to draw different types of borders inside a given rectangle. These borders can be used to outline an area or to give rectangles the effect of being recessed from or elevated above the surface of the screen.

The *sides*, *grays*, and *count* parameters determine how thick the border is and what gray levels are used to form it. This function uses the NSDivideRect function to take successive 1.0-unit-wide slices from the sides of the rectangle specified by the *sides* parameter. Each slice is drawn using the corresponding gray level from the *grays* parameter. This function makes and draws these slices *count* number of times. If you specify the same side more than once, the second slice is drawn inside the first.

The following example uses this function to draw a bezeled border consisting of a 1.0–unit-wide white line at the top and on the left side and a 1.0-unit-wide dark-gray line inside a 1.0–unit-wide black line on the other two sides. The resulting rectangle inside this border is then filled in using light gray.

```
NSRectEdge mySides[] = {NSMinYEdge, NSMaxXEdge, NSMaxYEdge, NSMinXEdge,
                        NSMinYEdge, NSMaxXEdge};
float myGrays[] = {NSBlack, NSBlack, NSWhite, NSWhite,
                        NSDarkGray, NSDarkGray};
NSRect aRect, clipRect; // Assume exists

aRect = NSDrawTiledRects(aRect, clipRect, mySides, myGrays, 6);
[[NSColor grayColor] set];
NSRectFill(aRect);
```

In the preceding example, *mySides* is an array that specifies sides of a rectangle; for example, `NSMinYEdge` selects the side parallel to the x axis with the smallest y coordinate value. *myGrays* is an array that specifies the successive gray levels to be used in drawing parts of the border.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphics.h`

NSDrawWhiteBezel

Draws a white-filled rectangle with a bezel border.

```
void NSDrawWhiteBezel (
   NSRect aRect,
   NSRect clipRect
);
```

Parameters

aRect

The bounding rectangle (in the current coordinate system) in which to draw. Only those parts of *aRect* that lie within the *clipRect* are actually drawn.

clipRect

The clipping rectangle to use during drawing.

Availability
Available in Mac OS X v10.0 and later.

See Also
`NSDrawTiledRects` (page 3978)

Related Sample Code
Sketch-112

Declared In
`NSGraphics.h`

NSDrawWindowBackground

Draws the window's default background pattern into the specified rectangle of the currently focused view.

```
void NSDrawWindowBackground (
   NSRect aRect
);
```

Parameters

aRect

The rectangle (in the current coordinate system) in which to draw the window's background pattern.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSGraphics.h

NSEnableScreenUpdates

Enables screen updates

```
void NSEnableScreenUpdates (void);
```

Discussion

Reenables, for all windows of a process, the flushing of drawing operations to the screen that was previously disabled by `NSDisableScreenUpdates` (page 3969). Successive calls to `NSDisableScreenUpdates` are placed on a stack and must be popped off that stack by matching calls to this function.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

AnimatedTableView

Declared In

NSGraphics.h

NSEraseRect

Erases the specified rect by filling it with white.

```
void NSEraseRect (
   NSRect aRect
);
```

Parameters

aRect

The rectangle (in the current coordinate system) defining the area to erase.

Discussion

This function fills the specified rectangle with white. It does not alter the current color.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz EB

Declared In
`NSGraphics.h`

NSEventMaskFromType

Returns the event mask for the specified type.

```
static NSUInteger NSEventMaskFromType (
   NSEventType type
);
```

Parameters

type

The event type whose mask you want to get.

Return Value
The event mask corresponding to the specified type. The returned mask is equivalent to the number 1 left-shifted by *type* bits.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer Matrix

Declared In
`NSEvent.h`

NSFrameRect

Draw a bordered rectangle.

```
void NSFrameRect (
   NSRect aRect
);
```

Parameters

aRect

The bounding rectangle (in the current coordinate system) in which to draw.

Discussion
Draws a frame around the inside of *aRect* in the current color and using the `NSCompositeCopy` compositing operation. The width is equal to 1.0 in the current coordinate system. Since the frame is drawn inside the rectangle, it will be visible even if drawing is clipped to the rectangle.

Because this function does not draw directly on the line, but rather inside it, it uses the current fill color (not stroke color) when drawing.

For a list of compositing operations and how you use them, see *Cocoa Drawing Guide*.

Availability
Available in Mac OS X v10.0 and later.

See Also
NSDrawTiledRects (page 3978)

Related Sample Code
CompositeLab
Cropped Image
FilterDemo
PDF Calendar
Sketch-112

Declared In
NSGraphics.h

NSFrameRectWithWidth

Draw a bordered rectangle.

```
void NSFrameRectWithWidth (
   NSRect aRect,
   CGFloat frameWidth
);
```

Parameters

aRect
The bounding rectangle (in the current coordinate system) in which to draw.

frameWidth
The width of the frame, specified in points.

Discussion
Draws a frame around the inside of *aRect* in the current color and using the NSCompositeCopy compositing operation. The width is equal to *frameWidth* in the current coordinate system. Since the frame is drawn inside the rectangle, it will be visible even if drawing is clipped to the rectangle.

Because this function does not draw directly on the line, but rather inside it, it uses the current fill color (not stroke color) when drawing.

For a list of compositing operations and how you use them, see *Cocoa Drawing Guide*.

Availability
Available in Mac OS X v10.0 and later.

See Also
NSDrawTiledRects (page 3978)

Related Sample Code
AnimatedTableView
Quartz Composer WWDC 2005 TextEdit
Rulers

Declared In
NSGraphics.h

NSFrameRectWithWidthUsingOperation

Draw a bordered rectangle using the specified compositing operation.

```
void NSFrameRectWithWidthUsingOperation (
   NSRect aRect,
   CGFloat frameWidth,
   NSCompositingOperation op
);
```

Parameters

aRect

The bounding rectangle (in the current coordinate system) in which to draw.

frameWidth

The width of the frame, specified in points.

op

The compositing operation to use when drawing the frame.

Discussion

Draws a frame around the inside of *aRect* in the current color, using the compositing operation *op*. The width is equal to *frameWidth* in the current coordinate system. Since the frame is drawn inside the rectangle, it will be visible even if drawing is clipped to the rectangle.

Because this function does not draw directly on the line, but rather inside it, it uses the current fill color (not stroke color) when drawing.

For a list of compositing operations and how you use them, see *Cocoa Drawing Guide*.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CocoaAUHost

EnhancedAudioBurn

PDF Annotation Editor

Declared In

NSGraphics.h

NSGetAlertPanel

Returns an alert panel.

```
id NSGetAlertPanel (
   NSString *title,
   NSString *msg,
   NSString *defaultButton,
   NSString *alternateButton,
   NSString *otherButton,
   ...
);
```

Discussion
Returns an `NSPanel` that can be used to set up a modal session. A modal session is useful for allowing the user to interrupt the program. During a modal session, you can perform activities while the panel is displayed and check at various points in your program whether the user has clicked one of the panel's buttons. The arguments for this function are the same as those for `NSRunAlertPanel` (page 3998), but unlike that function, no button is displayed if *`defaultButton`* is `nil`.

To set up a modal session, send the Application object `beginModalSessionForWindow:` (page 139) with the panel returned by `NSGetAlertPanel` as its argument. When you want to check if the user has clicked one of the panel's buttons, use `runModalSession:` (page 164). To end the modal session, use `endModalSession:` (page 146). When you're finished with the panel created by `NSGetAlertPanel`, you must free it by passing it to `NSReleaseAlertPanel` (page 3998).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPanel.h`

NSGetCriticalAlertPanel

Returns an alert panel to display a critical message.

```
id NSGetCriticalAlertPanel (
   NSString *title,
   NSString *msg,
   NSString *defaultButton,
   NSString *alternateButton,
   NSString *otherButton,
   ...
);
```

Discussion
Returns an `NSPanel` that can be used to set up a modal session. No button is displayed if *`defaultButton`* is `nil`. When you're finished with the panel created by this function, you must free it by passing it to `NSReleaseAlertPanel` (page 3998).

The arguments for this function are the same as those for the `NSGetAlertPanel` (page 3983). For more information on using a panel in a modal session, see `NSGetAlertPanel`.

The panel presented to the user is badged with a caution icon. Critical alerts should be used only as specified in the "Alerts" section of the Windows chapter of *Apple Human Interface Guidelines*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPanel.h

NSGetFileType

Returns a file type based on the passed pasteboard type. (Deprecated. The file contents pboard type allowed you to synthesize a pboard type for a file's contents based on the file's extension. Using the UTI of a file to represent its contents now replaces this functionality.)

```
NSString * NSGetFileType (
   NSString *pboardType
);
```

Discussion
This function is the inverse of both `NSCreateFileContentsPboardType` (page 3968) and `NSCreateFilenamePboardType` (page 3969). When passed a pasteboard type as returned by those functions, it returns the extension or filename from which the type was derived. It returns `nil` if *pboardType* isn't a pasteboard type created by those functions.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPasteboard.h

NSGetFileTypes

Returns an array of file types based on the passed pasteboard types. (Deprecated. The file contents pboard type allowed you to synthesize a pboard type for a file's contents based on the file's extension. Using the UTI of a file to represent its contents now replaces this functionality.)

```
NSArray * NSGetFileTypes (
   NSArray *pboardTypes
);
```

Discussion
Accepts a null-terminated array of pointers to pasteboard types and returns a null-terminated array of the unique extensions and filenames from the file content and filename types found in the input array. It returns `nil` if the input array contains no file content or filename types. The returned array is allocated and must be freed by the caller. The pointers in the return array point into strings passed in the input array.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPasteboard.h

NSGetInformationalAlertPanel

Returns an alert panel to display an informational message.

```
id NSGetInformationalAlertPanel (
   NSString *title,
   NSString *msg,
   NSString *defaultButton,
   NSString *alternateButton,
   NSString *otherButton,
   ...
);
```

Discussion
Returns an NSPanel that can be used to set up a modal session. No button is displayed if *defaultButton* is `nil`. When you're finished with the panel created by this function, you must free it by passing it to `NSReleaseAlertPanel` (page 3998).

The arguments for this function are the same as those for the `NSRunAlertPanel` (page 3998) function. For more information on using a panel in a modal session, see `NSGetAlertPanel` (page 3983).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSPanel.h`

NSGetWindowServerMemory

Returns the amount of memory being used by a context.

```
NSInteger NSGetWindowServerMemory (
   NSInteger context,
   NSInteger *virtualMemory,
   NSInteger *windowBackingMemory,
   NSString **windowDumpString
);
```

Discussion
Calculates the amount of memory being used at the moment by the given *context*. If `NULL` is passed for *context*, the current context is used. The amount of virtual memory used by the current context is returned in the int pointed to by *virtualMemory*; the amount of window backing store used by windows owned by the current context is returned in the int pointed to by *windowBackingMemory*. The sum of these two numbers is the amount of the memory that this context is responsible for.

Calculating these numbers takes some time to execute; thus, calling this function in normal operation is not recommended.

If `nil` is not passed in for *windowDumpStream*, the information returned is echoed to the specified stream. This fact can be useful for finding out more about which windows are using up your storage.

Normally, `NSGetWindowServerMemory` returns 0. If `NULL` is passed for *context* and there's no current display context, this function returns –1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphics.h`

NSHighlightRect

Highlights the specified rect by filling it with white.

```
void NSHighlightRect (
   NSRect aRect
);
```

Parameters

aRect

The bounding rectangle (in the current coordinate system) in which to draw.

Discussion

Highlights the rectangle referred to by *aRect*. Light gray becomes white, and white becomes light gray. This function must be called twice, once to highlight the rectangle and once to unhighlight it; the rectangle should not be left in its highlighted state. When not drawing on the screen, the compositing operation is replaced by one that fills the rectangle with light gray.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSGraphics.h

NSInterfaceStyleForKey

Returns an interface style value for the specified key and responder.

```
NSInterfaceStyle NSInterfaceStyleForKey (
   NSString *key,
   NSResponder *responder
);
```

Discussion

Used to determine an interface style based on a key and a responder, either of which may be `nil`. An `NSInterfaceStyle` (page 4009) value specifies the style in which an interface item, such as a button or a scroll bar, should be drawn. For example, a value of `NSMacintoshInterfaceStyle` indicates an item should be drawn in the Macintosh style. The values defined for `NSInterfaceStyle` are `NSNoInterfaceStyle`, `NSNextStepInterfaceStyle`, `NSWindows95InterfaceStyle`, and `NSMacintoshInterfaceStyle`. Note that this function never returns `NSNoInterfaceStyle`.

The interface style value returned by this function depends on several factors. If responder is not `nil` and if responder specifies an interface style other than `NSNoInterfaceStyle`, this function returns the responder's style, and *key* is ignored.

Otherwise, if *key* is not `nil` and there is an interface style for *key* specified by the defaults system, this function returns the interface style for *key* from the defaults system.

Finally, if *key* is `nil`, or if there is no interface style for *key* specified by the defaults system, this function returns the global interface style specified by the defaults system.

The defaults system allows an application to customize its behavior to match a user's preferences. You can read about the defaults system in the documentation for `NSUserDefaults`.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Declared In
`NSInterfaceStyle.h`

NSIsControllerMarker

Tests whether a given object is special marker object used for indicating the state of a selection in relation to a key.

```
BOOL NSIsControllerMarker (
   id object
);
```

Parameters

Term

Specify the object you want to check. This parameter can be `nil`.

Return Value
`YES` if the object is one of the designated controller markers or `NO` if it is not.

Discussion
This function helps you to create bindings between user interface elements and controller objects. The Application Kit predefines several special marker objects used as values for indicating selection state; currently these are `NSMultipleValuesMarker`, `NSNoSelectionMarker`, and `NSNotApplicableMarker`. These markers are typed as `id` and only exist for the purpose of indicating a state; they are never archived and cannot be used as object values in controls. You use this function to test whether a given object value is a marker, in which case it is not directly assignable to the object that is bound. This check is important, especially since additional markers may be added in the future.

See the `NSKeyValueBinding.h` header file for further details.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
CoreRecipes

Declared In
`NSKeyValueBinding.h`

NSNumberOfColorComponents

Returns the number of color components in the specified color space.

```
NSInteger NSNumberOfColorComponents (
   NSString *colorSpaceName
);
```

Discussion
Returns the number of color components in the color space whose name is provided by `colorSpaceName`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSGraphics.h

NSOpenGLGetOption

Returns global OpenGL options.

```
void NSOpenGLGetOption (
   NSOpenGLGlobalOption pname,
   GLint *param
);
```

Discussion
Returns in *param* the value of the global OpenGL parameter *pname*. The available options are enumerated by the NSOpenGLGlobalOption (page 4011) type.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSOpenGL.h

NSOpenGLGetVersion

Returns the NSOpenGL version numbers.

```
void NSOpenGLGetVersion (
   GLint *major,
   GLint *minor
);
```

Discussion
Returns by reference the major and minor version numbers of the NSOpenGL implementation. This function is not the same as the OpenGL version.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSOpenGL.h

NSOpenGLSetOption

Sets global OpenGL options.

```
void NSOpenGLSetOption (
   NSOpenGLGlobalOption pname,
   GLint param
);
```

Discussion
Sets the value of the global OpenGL parameter *pname* to *param*. The available options are enumerated by the `NSOpenGLGlobalOption` (page 4011) type.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOpenGL.h`

NSPerformService

Programmatically invokes a Services menu service.

```
BOOL NSPerformService (
   NSString *itemName,
   NSPasteboard *pboard
);
```

Parameters

itemName

Specifies a Services menu item, in any language. If the requested service is from a submenu of the Services menu, the value must contain a slash (for example, “Mail/Selection”).

pboard

The pasteboard containing the data required by the service. This data must be present for the service to succeed. On output, this pasteboard contains the data returned by the service provider.

Return Value
`YES` if the service was successfully performed or `NO` if it was not.

Discussion
Use this function to programmatically invoke a service found in the application’s Services menu.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
PhotoSearch

Declared In
`NSApplication.h`

NSPlanarFromDepth

Returns whether the specified window depth is planar.

```
BOOL NSPlanarFromDepth (
   NSWindowDepth depth
);
```

Discussion
Returns `YES` if the specified window *`depth`* is planar and `NO` if it is not.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphics.h`

NSReadPixel

Reads the color of the pixel at the specified location.

```
NSColor * NSReadPixel (
   NSPoint passedPoint
);
```

Parameters

`passedPoint`

The pixel location to read, specified in the current coordinate system.

Return Value
The color of the pixel at the specified location.

Discussion
Because the *`passedPoint`* parameter is relative to the current coordinate system, if you wish to read a pixel from a specific view, you must convert points in the view's coordinate system to the current coordinate system before calling this function. Alternatively, you can lock focus on the view and then specify the pixel coordinate in the view's coordinate system.

When mapping the specified point to pixel boundaries, this method rounds to the nearest pixel. For more information on how coordinate points map to the underlying pixels, see Coordinate Systems and Transforms in *Cocoa Drawing Guide*.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Color Sampler
Monochrome Image

Declared In
`NSGraphics.h`

NSRectClip

Modifies the current clipping path by intersecting it with the passed rect.

```
void NSRectClip (
   NSRect aRect
);
```

Parameters

aRect

The rectangle to intersect with the current clipping rectangle.

Discussion
This function modifies the clipping path permanently. If you need to undo this modification later, you should save the current graphics state before calling this function and restore it once you are done.

A side effect of this function is that it clears the current Quartz 2D drawing path information. If you used Quartz 2D functions to create a drawing path in the current context, and you want to save that path information and use it later, you should transfer it to a `CGPathRef` opaque type before calling this function. If you are using only Cocoa to do your drawing, this behavior should not affect you.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Aperture Edit Plugin - Borders & Titles
Rulers
SampleRaster

Declared In
`NSGraphics.h`

NSRectClipList

Modifies the current clipping path by intersecting it with the passed rect.

```
void NSRectClipList (
   const NSRect *rects,
   NSInteger count
);
```

Parameters

rects

A pointer to an array of `NSRect` structures, which are combined and intersected with the current clipping path.

count

The number of rectangles in *rects*.

Discussion
This function modifies the clipping path permanently by generating a graphical union of the specified rectangles and then intersecting that union with the current clipping path. If you need to undo this modification later, you should save the current graphics state before calling this function and restore it once you are done.

A side effect of this function is that it clears the current Quartz 2D drawing path information. If you used Quartz 2D functions to create a drawing path in the current context, and you want to save that path information and use it later, you should transfer it to a `CGPathRef` opaque type before calling this function. If you are using only Cocoa to do your drawing, this behavior should not affect you.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSGraphics.h

NSRectFill

Fills the passed rectangle with the current color.

```
void NSRectFill (
   NSRect aRect
);
```

Parameters

aRect

The bounding rectangle (in the current coordinate system) in which to draw.

Discussion
Fills *aRect* with the current color using the compositing mode `NSCompositeCopy` (page 1376), which fills with the current color by copying the RGBA values. Use `NSRectFillUsingOperation` (page 3997) to fill specifying a compositing mode.

For a list of compositing operations and how you use them, see *Cocoa Drawing Guide*.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaSlides
FilterDemo
Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility
Sketch-112

Declared In
NSGraphics.h

NSRectFillList

Fills the rectangles in the passed list with the current fill color.

```
void NSRectFillList (
   const NSRect *rects,
   NSInteger count
);
```

Parameters

rects

A pointer to an array of `NSRect` structures representing the rectangles to fill.

count

The number of rectangles in *rects*.

Discussion
Fills the specified rectangles with the current fill color using the compositing mode `NSCompositeCopy`.

For a list of compositing operations and how you use them, see *Cocoa Drawing Guide*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphics.h`

NSRectFillListUsingOperation

Fills the rectangles in a list using the current fill color and specified compositing operation.

```
void NSRectFillListUsingOperation (
   const NSRect *rects,
   NSInteger count,
   NSCompositingOperation op
);
```

Parameters

rects
: A pointer to an array of `NSRect` structures representing the rectangles to fill.

count
: The number of rectangles in the *rects* parameter.

op
: The compositing operation to use when filling the rectangles.

Discussion
Fills a list of *count* rectangles with the current fill color, using the compositing operation *op*. For example, specifying `NSCompositeSourceOver` (page 1376) will blend with what's already been drawn.

For a list of compositing operations and how you use them, see *Cocoa Drawing Guide*.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
iChatTheater

Declared In
`NSGraphics.h`

NSRectFillListWithColors

Fills the rectangles in the passed list with the passed list of colors.

```
void NSRectFillListWithColors (
   const NSRect *rects,
   NSColor **colors,
   NSInteger num
);
```

Parameters

rects

A pointer to an array of `NSRect` structures representing the rectangles to fill.

colors

A pointer to an array of `NSColor` objects. The number of color objects in this parameter must equal the number of rectangles in the *rects* parameter.

num

The number of rectangles in the *rects* parameter.

Discussion

Takes a list of *num* rectangles and a matching list of color objects. The first rectangle is filled with the first color, the second rectangle with the second color, and so on. There must be an equal number of rectangles and color values. The rectangles are composited using the `NSCompositeCopy` (page 1376) operator and the order in which the rectangles are filled cannot be guaranteed; therefore, overlapping rectangles may not draw as expected. This function alters the current color of the current graphics state, setting it unpredictably to one of the values passed in *colors*.

For a list of compositing operations and how you use them, see *Cocoa Drawing Guide*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSGraphics.h`

NSRectFillListWithColorsUsingOperation

Fills the rectangles in a list using the specified colors and compositing operation.

```
void NSRectFillListWithColorsUsingOperation (
   const NSRect *rects,
   NSColor **colors,
   NSInteger num,
   NSCompositingOperation op
);
```

Parameters

rects

A pointer to an array of `NSRect` structures representing the rectangles to fill.

colors

A pointer to an array of `NSColor` objects. The number of color objects in this parameter must equal the number of rectangles in the *rects* parameter.

num

The number of rectangles in the *rects* parameter.

op

The compositing operation to use when filling the rectangles.

Discussion
Takes a list of *num* rectangles and a matching list of color values. The first rectangle is filled with the first color, the second rectangle with the second color, and so on. There must be an equal number of rectangles and color values. Each fill operation is performed using the compositing operation *op*. The rectangles should not overlap; the order in which they are filled cannot be guaranteed. This function alters the current color of the current graphics state, setting it unpredictably to one of the values passed in *colors*.

For a list of compositing operations and how you use them, see *Cocoa Drawing Guide*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSGraphics.h

NSRectFillListWithGrays

Fills the rectangles in the passed list with the passed list of grays.

```
void NSRectFillListWithGrays (
   const NSRect *rects,
   const CGFloat *grays,
   NSInteger num
);
```

Parameters

rects
A pointer to an array of NSRect structures representing the rectangles to fill.

grays
A pointer to an array of floating-point values in the range 0.0 to 1.0, where 0.0 represents absolute black and 1.0 represents absolute white and numbers in between are varying levels of gray. Values outside this range are clamped to 0.0 or 1.0.

num
The number of rectangles in the *rects* parameter.

Discussion
Takes a list of *num* rectangles and a matching list of gray values. The first rectangle is filled with the first gray, the second rectangle with the second gray, and so on. There must be an equal number of rectangles and gray values. The rectangles are composited using the NSCompositeCopy (page 1376) operator and the order in which the rectangles are filled cannot be guaranteed; therefore, overlapping rectangles may not draw as expected. This function alters the current color of the current graphics state, setting it unpredictably to one of the values passed in *grays*.

For a list of compositing operations and how you use them, see *Cocoa Drawing Guide*.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSGraphics.h

NSRectFillUsingOperation

Fills a rectangle using the current fill color and the specified compositing operation.

```
void NSRectFillUsingOperation (
   NSRect aRect,
   NSCompositingOperation op
);
```

Parameters

aRect

The rectangle to fill with the current fill color.

op

The compositing operation to use when filling the rectangle.

Discussion

For a list of compositing operations and how you use them, see *Cocoa Drawing Guide*.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ComplexBrowser

Cropped Image

RGB Image

RGB ValueTransformers

Tinted Image

Declared In

`NSGraphics.h`

NSRegisterServicesProvider

Registers a service provider.

```
void NSRegisterServicesProvider (
   id provider,
   NSString *name
);
```

Parameters

provider

The object providing the service you want to register.

name

The unique name to associate with the service. This string is used to advertise the service to interested clients.

Discussion

Use this function to register custom services not directly related to your application.

You should not use this function to register the services provided by your application. For your application's services, you should use the `setServicesProvider:` (page 171) method of `NSApplication`, passing a non-`nil` argument.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSApplication.h

NSReleaseAlertPanel

Releases an alert panel.

```
void NSReleaseAlertPanel (
   id panel
);
```

Discussion
When you're finished with a panel created by a function such as `NSGetAlertPanel` (page 3983), `NSGetCriticalAlertPanel` (page 3984), or `NSGetInformationalAlertPanel` (page 3985), you must free it by passing it to this function.

Note that the alert panel may not be deallocated immediately because it may have internal references that are released in a deferred way. You should not make the assumption that the alert panel is immediately removed from the application window list.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPanel.h

NSRunAlertPanel

Creates an alert panel.

```
NSInteger NSRunAlertPanel (
   NSString *title,
   NSString *msg,
   NSString *defaultButton,
   NSString *alternateButton,
   NSString *otherButton,
   ...
);
```

Discussion
Creates and runs an alert panel (or dialog) with the title of *title*, the text of *msg*, and buttons with titles of *defaultButton*, *alternateButton*, and *otherButton*. See the description of `NSBeginAlertSheet` (page 3962) for information on layout of buttons, default parameters, and possible return values. `NSRunAlertPanel` runs the panel in a modal event loop.

A Command-D key equivalent for the "Don't Save" button is provided, if one is found. The button titles are searched for the localized value for "Don't Save." If a match is found, that button is assigned a Command-D key equivalent, provided it is not the default button.

If you create a modal panel using `runModalForWindow:` (page 163) or `beginSheet:modalForWindow:modalDelegate:didEndSelector:contextInfo:` (page 140), you can assign the key equivalent yourself, using `setKeyEquivalent:` (page 483) and `setKeyEquivalentModifierMask:` (page 484).

This function not only creates the panel; it also puts the panel onscreen and runs it using the `runModalForWindow:` (page 163) method defined in the `NSApplication` class. This method sets up a modal event loop that causes the panel to remain onscreen until the user clicks one of its buttons. This function then removes the panel from the screen list and returns a value that indicates which of the three buttons the user clicked. For efficiency, this function creates the panel the first time it's called and reuses it on subsequent calls, reconfiguring it if necessary.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaSpeechSynthesisExample
Quartz Composer WWDC 2005 TextEdit
SimpleCocoaApp
TextLinks
WhackedTV

Declared In
`NSPanel.h`

NSRunCriticalAlertPanel

Creates and runs a critical alert panel.

```
NSInteger NSRunCriticalAlertPanel (
   NSString *title,
   NSString *msg,
   NSString *defaultButton,
   NSString *alternateButton,
   NSString *otherButton,
   ...
);
```

Discussion
Creates a critical alert panel that warns the user of some critical consequence of a requested action; the panel lets the user cancel the action and may allow the user to modify the action. It then runs the panel in a modal event loop.

The panel presented to the user is badged with a caution icon. Critical alerts should be used only as specified in the "Alerts" section of the Windows chapter of *Apple Human Interface Guidelines*.

The arguments for this function are the same as those for `NSRunAlertPanel` (page 3998).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CIColorTracking
GLSL Showpiece Lite

GLUT
iChatStatusFromApplication
NewsReader

Declared In
NSPanel.h

NSRunInformationalAlertPanel

Creates and runs an informational alert panel.

```
NSInteger NSRunInformationalAlertPanel (
   NSString *title,
   NSString *msg,
   NSString *defaultButton,
   NSString *alternateButton,
   NSString *otherButton,
   ...
);
```

Discussion
Creates an informational alert panel that provides information related to a requested action. It then runs the panel in a modal event loop.

The arguments for this function are the same as those for `NSRunAlertPanel` (page 3998).

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSPanel.h

NSSetFocusRingStyle

Specifies how a focus ring will be drawn.

```
void NSSetFocusRingStyle (
   NSFocusRingPlacement placement
);
```

Parameters

placement
Specifies how you want the focus ring to be drawn.

Discussion
Use `NSFocusRingAbove` to draw the focus ring over an image, use `NSFocusRingBelow` to draw the focus ring under text, and use `NSFocusRingOnly` if you don't have an image or text. For the `NSFocusRingOnly` case, fills a shape to add the focus ring around the shape.

Note that the focus ring may actually be drawn outside the view but will be clipped to any clipping superview or the window content view.

Availability
Available in Mac OS X v10.1 and later.

Related Sample Code
ClockControl
Dicey
TrackBall

Declared In
`NSGraphics.h`

NSSetShowsServicesMenuItem

Specifies whether an item should be included in Services menus.

```
NSInteger NSSetShowsServicesMenuItem (
   NSString *itemName,
   BOOL enabled
);
```

Discussion
Deprecated. This function simply returns 0.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSShowAnimationEffect

Runs a system animation effect.

```
void NSShowAnimationEffect (
   NSAnimationEffect animationEffect,
   NSPoint centerLocation,
   NSSize size,
   id animationDelegate,
   SEL didEndSelector,
   void *contextInfo
);
```

Parameters

animationEffect
The type of animation you want to apply.

centerLocation
The location at which to show the animated image, specified in screen coordinates. The animation is centered on the point you specify.

size
The desired size of the animated image. Specify `NSZeroSize` to perform the animation at the default size.

animationDelegate
The object to notify when the animation completes. Specify `nil` if you do not need to be notified when the animation completes.

didEndSelector

The selector of *animationDelegate* to call when the animation completes. Specify `nil` if you do not need to be notified when the animation completes. If you specify a selector, the corresponding method should have the following signature:

```
- (void)animationEffectDidEnd:(void *)contextInfo;
```

contextInfo

A pointer to any optional information you want passed as a parameter to the selector in the *didEndSelector* parameter.

Discussion
This function runs one of the standard system animation effects, which includes display and sound. For example, you can use this function to display the puff of smoke effect. For a complete list of animation effects, see `NSAnimationEffect` (page 4007).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSGraphics.h`

NSShowsServicesMenuItem

Specifies whether a Services menu item is currently enabled.

```
BOOL NSShowsServicesMenuItem (
   NSString *itemName
);
```

Discussion
Deprecated. This function simply returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSUnregisterServicesProvider

Unregisters a service provider.

```
void NSUnregisterServicesProvider(NSString *name);
```

Parameters

name

The name of the service you want to unregister.

Discussion
Use this function to unregister custom services not directly related to your application.

You should not use this function to unregister the services provided by your application. For your application's services, you should use the `setServicesProvider:` (page 171) method of `NSApplication`, passing a `nil` argument.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSApplication.h

NSUpdateDynamicServices

Causes the services information for the system to be updated.

```
void NSUpdateDynamicServices (void);
```

Discussion
Used by a service-providing application to reregister the services it is willing to provide. To do this, you create a bundle with the extension ".service" and place it in the application's path or ~/Library/Services. The content of the bundle is identical to a normal service bundle. You then call this function.

It is only necessary to call this function if your program adds dynamic services to the system.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
SpotlightFortunes

Declared In
NSApplication.h

NSWindowList

Gets information about onscreen windows.

```
void NSWindowList (
   NSInteger size,
   NSInteger list[]
);
```

Discussion
Provides an ordered list of all onscreen windows. It fills *list* with up to *size* window numbers; the order of windows in the array is the same as their order in the window server's screen list (their front-to-back order on the screen). Use the count obtained by NSCountWindows (page 3968) to specify the size of the array for this function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSGraphics.h

NSWindowListForContext

Gets information about an application's onscreen windows.

```
void NSWindowListForContext (
   NSInteger context,
   NSInteger size,
   NSInteger list[]
);
```

Discussion
Provides an ordered list of onscreen windows for a particular application, identified by *context*, which is a window server connection ID. It fills *list* with up to *size* window numbers; the order of windows in the array is the same as their order in the window server's screen list (their front-to-back order on the screen). Use the count obtained by the `NSCountWindowsForContext` (page 3968) function to specify the size of the array for this function.

Use of this function is discouraged as it may be deprecated in a future release.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSGraphics.h`

Data Types

PART IV

Data Types

Application Kit Data Types Reference

Framework: AppKit/AppKit.h

Overview

This document describes the data types defined in the Application Kit framework and not described in a document for an individual class.

Data Types

NSAnimationEffect

This type defines the standard system animation effects, which include both display and sound.

```
typedef enum _NSAnimationEffect {
    NSAnimationEffectDisappearingItemDefault = 0,
    NSAnimationEffectPoof = 10
} NSAnimationEffect;
```

Constants

`NSAnimationEffectDisappearingItemDefault`

The default effect.

Available in Mac OS X v10.3 and later.

Declared in `NSGraphics.h`.

`NSAnimationEffectPoof`

An effect showing a puff of smoke.

Available in Mac OS X v10.3 and later.

Declared in `NSGraphics.h`.

Discussion

These effects are used to indicate that an item was removed from a collection, such as a toolbar, without deleting the underlying data. See `NSShowAnimationEffect` (page 4001).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`NSGraphics.h`

NSBrowserAuxiliaryOpaque

A private data structure used internally by `NSBrowser`.

```
typedef struct NSBrowserAuxiliary NSBrowserAuxiliaryOpaque;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSBrowser.h`

NSColorListAuxiliaryOpaque

A private data structure used internally by `NSColorList`.

```
typedef struct NSColorListAuxiliary NSColorListAuxiliaryOpaque;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSColorList.h`

NSFocusRingPlacement

The focus ring style indicates how the focus ring will be drawn.

```
typedef enum {
    NSFocusRingOnly  = 0,
    NSFocusRingBelow = 1,
    NSFocusRingAbove = 2
} NSFocusRingPlacement;
```

Constants

`NSFocusRingAbove`
Use `NSFocusRingAbove` to draw over an image.
Fill a shape to add the focus ring around the shape.
Available in Mac OS X v10.1 and later.
Declared in `NSGraphics.h`.

`NSFocusRingBelow`
Use `NSFocusRingBelow` to draw the focus ring under text.
Available in Mac OS X v10.1 and later.
Declared in `NSGraphics.h`.

`NSFocusRingOnly`
Use `NSFocusRingOnly` if you don't have an image or text.
Available in Mac OS X v10.1 and later.
Declared in `NSGraphics.h`.

Availability
Available in Mac OS X v10.1 and later.

Declared In
NSGraphics.h

NSFocusRingType

The focus ring type is used by NSView and NSCell to configure if and how a control should draw its focus ring.

```
typedef enum _NSFocusRingType {
    NSFocusRingTypeDefault  = 0,
    NSFocusRingTypeNone     = 1,
    NSFocusRingTypeExterior = 2
} NSFocusRingType;
```

Constants

NSFocusRingTypeDefault
The default focus ring type for NSView or NSCell.
Available in Mac OS X v10.3 and later.
Declared in NSGraphics.h.

NSFocusRingTypeNone
No focus ring. If you set the focus ring type to this value, NSView and NSCell will not draw any focus ring.
Available in Mac OS X v10.3 and later.
Declared in NSGraphics.h.

NSFocusRingTypeExterior
The standard Aqua focus ring.
Available in Mac OS X v10.3 and later.
Declared in NSGraphics.h.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSGraphics.h

NSInterfaceStyle

These constants are used in NSResponder's interfaceStyle (page 2158) method.

```
typedef enum {
    NSNoInterfaceStyle        = 0,
    NSNextStepInterfaceStyle  = 1,
    NSWindows95InterfaceStyle = 2,
    NSMacintoshInterfaceStyle = 3
} NSInterfaceStyle;
```

Constants

`NSNoInterfaceStyle`
The default interface style.
Available in Mac OS X v10.0 and later.
Declared in `NSInterfaceStyle.h`.

`NSNextStepInterfaceStyle`
The NextStep interface style.
Available in Mac OS X v10.0 and later.
Declared in `NSInterfaceStyle.h`.

`NSWindows95InterfaceStyle`
The Windows 95 interface style.
Available in Mac OS X v10.0 and later.
Declared in `NSInterfaceStyle.h`.

`NSMacintoshInterfaceStyle`
The Macintosh interface style.
Available in Mac OS X v10.0 and later.
Declared in `NSInterfaceStyle.h`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSInterfaceStyle.h`

NSModalSession

Variables of type `NSModalSession` point to information used by the system between `NSApplication`'s `beginModalSessionForWindow:` (page 139) and `endModalSession:` (page 146) messages.

```
typedef struct _NSModalSession *NSModalSession;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSApplication.h`

NSOpenGLContextAuxiliary

A private data structure used by `NSOpenGLContext`.

```
typedef struct _CGLContextObject NSOpenGLContextAuxiliary;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSOpenGL.h

NSOpenGLGlobalOption

These constants are option names for NSOpenGLSetOption (page 3989) and NSOpenGLGetOption (page 3989).

```
typedef enum {
    NSOpenGLGOFormatCacheSize  = 501,
    NSOpenGLGOClearFormatCache = 502,
    NSOpenGLGORetainRenderers  = 503,
    NSOpenGLGOResetLibrary     = 504
} NSOpenGLGlobalOption;
```

Constants

NSOpenGLGOFormatCacheSize
Sets the size of the pixel format cache.
Available in Mac OS X v10.0 and later.
Declared in NSOpenGL.h.

NSOpenGLGOClearFormatCache
Resets the pixel format cache if true.
Available in Mac OS X v10.0 and later.
Declared in NSOpenGL.h.

NSOpenGLGORetainRenderers
Whether to retain loaded renderers in memory.
Available in Mac OS X v10.0 and later.
Declared in NSOpenGL.h.

NSOpenGLGOResetLibrary
Does a soft reset of the CGL library if true.
Available in Mac OS X v10.0 and later.
Declared in NSOpenGL.h.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSOpenGL.h

NSOpenGLPixelFormatAuxiliary

A private data structure used by NSOpenGLPixelFormat.

```
typedef struct _CGLPixelFormatObject NSOpenGLPixelFormatAuxiliary;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSOpenGL.h`

NSSavePanelAuxiliaryOpaque

A private data structure used internally by `NSSavePanel`.

```
typedef struct NSSavePanelAuxiliary NSSavePanelAuxiliaryOpaque;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSSavePanel.h`

NSScreenAuxiliaryOpaque

A private data structure used internally by `NSScreen`.

```
typedef struct NSScreenAuxiliary NSScreenAuxiliaryOpaque;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSScreen.h`

NSTabViewItemAuxiliaryOpaque

A private data structure used by `NSTabViewItem`.

```
typedef struct NSTabViewItemAuxiliary NSTabViewItemAuxiliaryOpaque;
```

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSTabViewItem.h`

NSTypesetterGlyphInfo

This type is a caching structure used by NSSimpleHorizontalTypesetter.

```
typedef struct _NSTypesetterGlyphInfo {
    NSPoint curLocation;
    float extent;
    float belowBaseline;
    float aboveBaseline;
    unsigned glyphCharacterIndex;
    NSFont *font;
    NSSize attachmentSize;
    struct {
        BOOL defaultPositioning:1;
        BOOL dontShow:1;
        BOOL isAttachment:1;
    } _giflags;
} NSTypesetterGlyphInfo;
```

Fields

`curLocation`
Location (relative to the baseline) for laying this glyph out.

`extent`
Required space from `curLocation` to lay this glyph out; –1.0 if not set.

`belowBaseline`
Distance from baseline to bottom of the line fragment required for all the glyphs so far, including this one (positive if baseline is above the bottom of the line fragment).

`aboveBaseline`
Distance from baseline to top of the line fragment required for all the glyphs so far, including this one (positive if baseline is below the top of the line fragment).

`glyphCharacterIndex`
Character index.

`font`
Font.

`attachmentSize`
Size of the character if it's an attachment; otherwise meaningless.

`defaultPositioning`
This block needs to be "show"ed.

`dontShow`
Don't show this glyph.

`isAttachment`
Whether the glyph is an attachment.

Availability
Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared In
`NSSimpleHorizontalTypesetter.h`

Constants

PART V

Constants

Application Kit Constants Reference

Framework: /System/Library/Frameworks/AppKit.framework

Declared in AppKit/AppKitDefines.h
AppKit/NSErrors.h
AppKit/AppKitErrors.h
AppKit/NSNibDeclarations.h
AppKit/NSGraphics.h

Overview

This document describes the constants defined in the Application Kit framework that are not defined in, or are not described in, a document for an individual class.

See *Application Kit Data Types Reference* for descriptions of other constants defined in enumerations.

Constants

There are three types of constant in this document: global variables, errors, and exceptions.

Global Variables

Color Space Names

Color-space names designate predefined color spaces.

```
NSString *NSCalibratedWhiteColorSpace;
NSString *NSCalibratedBlackColorSpace;
NSString *NSCalibratedRGBColorSpace;
NSString *NSDeviceWhiteColorSpace;
NSString *NSDeviceBlackColorSpace;
NSString *NSDeviceRGBColorSpace;
NSString *NSDeviceCMYKColorSpace;
NSString *NSNamedColorSpace;
NSString *NSPatternColorSpace;
NSString *NSCustomColorSpace;
```

Constants

`NSCalibratedWhiteColorSpace`

Calibrated color space with white and alpha components (pure white is 1.0)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSCalibratedBlackColorSpace`

Calibrated color space with black and alpha components (pure black is 1.0)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

Declared in `NSGraphics.h`.

`NSCalibratedRGBColorSpace`

Calibrated color space with red, green, blue, and alpha components.

You can also create a color with HSB (hue, saturation, brightness) and alpha components and can extract these components.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSDeviceWhiteColorSpace`

Device-dependent color space with white and alpha components (pure white is 1.0)

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSDeviceBlackColorSpace`

Device-dependent color space with black and alpha components (pure black is 1.0)

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

Declared in `NSGraphics.h`.

`NSDeviceRGBColorSpace`

Device-dependent color space with red, green, blue, and alpha components.

You can also create a color with HSB (hue, saturation, brightness) and alpha components and can extract these components.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSDeviceCMYKColorSpace`

Device-dependent color space with cyan, magenta, yellow, black, and alpha components

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSNamedColorSpace`
: Catalog name and color name components

  The components of this color space are indexes into lists or catalogs of prepared colors. The catalogs of named colors come with lookup tables that are able to generate the correct color on a given device.

  Available in Mac OS X v10.0 and later.

  Declared in `NSGraphics.h`.

`NSPatternColorSpace`
: Pattern image (tiled)

  Identifies a pattern color space, which is simply an image that is repeated over and over again in a tiled pattern.

  Available in Mac OS X v10.0 and later.

  Declared in `NSGraphics.h`.

`NSCustomColorSpace`
: Custom `NSColorSpace` object and floating-point components describing a color in that space

  A custom color-space object represents a color space that is not necessarily predefined by the Application Kit. See “Working With Color Spaces” for information on creating custom color-space objects.

  Available in Mac OS X v10.0 and later.

  Declared in `NSGraphics.h`.

Discussion
You can use a color-space name in certain methods of `NSColor` that create or convert color objects. The name identifies the color space to be used for the operation.

Declared In
`NSGraphics.h`

Grayscale Values

These constants are the standard gray values for the 2-bit deep grayscale color space.

```
const float NSWhite;
const float NSLightGray;
const float NSDarkGray;
const float NSBlack;
```

Constants
`NSWhite`
: A constant that specifies the white shade in the 2-bit deep grayscale color space.

  Available in Mac OS X v10.0 and later.

  Declared in `NSGraphics.h`.

`NSLightGray`
: A constant that specifies the light gray shade in the 2-bit deep grayscale color space.

  Available in Mac OS X v10.0 and later.

  Declared in `NSGraphics.h`.

`NSDarkGray`
A constant that specifies the dark gray shade in the 2-bit deep grayscale color space.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

`NSBlack`
A constant that specifies the black shade in the 2-bit deep grayscale color space.

Available in Mac OS X v10.0 and later.

Declared in `NSGraphics.h`.

NSInterfaceStyleDefault

`NSInterfaceStyleDefault` can be used to override the platform's default interface style.

```
NSString *NSInterfaceStyleDefault;
```

Constants
`NSInterfaceStyleDefault`
For more information, see the function `NSInterfaceStyleForKey` (page 3987).

Available in Mac OS X v10.0 and later.

Declared in `NSInterfaceStyle.h`.

Interface Builder Constants

Type qualifiers used by Interface Builder to synchronize with Xcode. For more information, see Communicating With Objects in *Cocoa Fundamentals Guide*.

```
#define IBAction void
#define IBOutlet
```

Constants
`IBAction`
Type qualifier used by Interface Builder to synchronize actions added programmatically with its internal list of action methods defined for a project.

Available in Mac OS X v10.0 and later.

Declared in `NSNibDeclarations.h`.

`IBOutlet`
Identifier used to qualify an instance-variable declaration so that Interface Builder can synchronize the display and connection of outlets with Xcode.

Available in Mac OS X v10.0 and later.

Declared in `NSNibDeclarations.h`.

NSWindow—Sizes

Obsolete constant values. Do not use.

```
NSSize NSIconSize;
NSSize NSTokenSize;
```

Constants

`NSIconSize`

Obsolete constant values. Do not use.

Available in Mac OS X v10.0 through Mac OS X v10.3.

Declared in `NSWindow.h`.

`NSTokenSize`

Obsolete constant values. Do not use.

Available in Mac OS X v10.0 through Mac OS X v10.3.

Declared in `NSWindow.h`.

Errors

Attributed String Errors

These constants represent errors generated by `NSAttributedString`.

```
enum {
    NSTextReadInapplicableDocumentTypeError = 65806,
    NSTextWriteInapplicableDocumentTypeError = 66062,
    NSTextReadWriteErrorMinimum = 65792,
    NSTextReadWriteErrorMaximum = 66303
};
```

Constants

`NSTextReadInapplicableDocumentTypeError`

Indicates a problem reading data with the specified format.

Available in Mac OS X v10.4 and later.

Declared in `AppKitErrors.h`.

`NSTextWriteInapplicableDocumentTypeError`

Indicates a problem writing data of the specified format.

Available in Mac OS X v10.4 and later.

Declared in `AppKitErrors.h`.

`NSTextReadWriteErrorMinimum`

The beginning of a range of error codes reserved for future use.

Available in Mac OS X v10.4 and later.

Declared in `AppKitErrors.h`.

`NSTextReadWriteErrorMaximum`

The end of a range of error codes reserved for future use.

Available in Mac OS X v10.4 and later.

Declared in `AppKitErrors.h`.

Discussion

These constants are returned in an `NSError` object.

Services Error Codes

These constants represent errors returned by application services.

```
enum {
    NSServiceApplicationNotFoundError = 66560,
    NSServiceApplicationLaunchFailedError = 66561,
    NSServiceRequestTimedOutError = 66562,
    NSServiceInvalidPasteboardDataError = 66563,
    NSServiceMalformedServiceDictionaryError = 66564,
    NSServiceMiscellaneousError = 66800,
    NSServiceErrorMinimum = 66560,
    NSServiceErrorMaximum = 66817
};
```

Constants

`NSServiceApplicationNotFoundError`
The service provider could not be found.

Available in Mac OS X v10.5 and later.

Declared in `AppKitErrors.h`.

`NSServiceApplicationLaunchFailedError`
The service providing application could not be launched. This will typically contain an underlying error with an Launch Services error code.

Available in Mac OS X v10.5 and later.

Declared in `AppKitErrors.h`.

`NSServiceRequestTimedOutError`
The service providing application did not open its service listening port in time, or the app didn't respond to the request in time; see the Console log to figure out which (the errors are typically reported the same way to the user).

Available in Mac OS X v10.5 and later.

Declared in `AppKitErrors.h`.

`NSServiceInvalidPasteboardDataError`
The service providing app did not return a pasteboard with any of the promised types, or we couldn't write the data from the pasteboard to the object receiving the returned data.

Available in Mac OS X v10.5 and later.

Declared in `AppKitErrors.h`.

`NSServiceMalformedServiceDictionaryError`
The service dictionary did not contain the necessary keys. Messages will typically be logged to the console giving more details.

Available in Mac OS X v10.5 and later.

Declared in `AppKitErrors.h`.

`NSServiceMiscellaneousError`
Other errors, representing programmatic mistakes in the service consuming application. These show a generic error message to the user.

Available in Mac OS X v10.5 and later.

Declared in `AppKitErrors.h`.

`NSServiceErrorMinimum`

Inclusive service error range, for checking future error codes.

Available in Mac OS X v10.5 and later.

Declared in `AppKitErrors.h`.

`NSServiceErrorMaximum`

Inclusive service error range, for checking future error codes.

Available in Mac OS X v10.5 and later.

Declared in `AppKitErrors.h`.

Exceptions

Application Kit Exception Names

These constants name the exceptions that the Application Kit can raise.

```
NSString *NSTextLineTooLongException;
NSString *NSTextNoSelectionException;
NSString *NSWordTablesWriteException;
NSString *NSWordTablesReadException;
NSString *NSTextReadException;
NSString *NSTextWriteException;
NSString *NSPasteboardCommunicationException;
NSString *NSPrintingCommunicationException;
NSString *NSAbortModalException;
NSString *NSAbortPrintingException;
NSString *NSIllegalSelectorException;
NSString *NSAppKitVirtualMemoryException;
NSString *NSBadRTFDirectiveException;
NSString *NSBadRTFFontTableException;
NSString *NSBadRTFStyleSheetException;
NSString *NSTypedStreamVersionException;
NSString *NSTIFFException;
NSString *NSPrintPackageException;
NSString *NSBadRTFColorTableException;
NSString *NSDraggingException;
NSString *NSColorListIOException;
NSString *NSColorListNotEditableException;
NSString *NSBadBitmapParametersException;
NSString *NSWindowServerCommunicationException;
NSString *NSFontUnavailableException;
NSString *NSPPDIncludeNotFoundException;
NSString *NSPPDParseException;
NSString *NSPPDIncludeStackOverflowException;
NSString *NSPPDIncludeStackUnderflowException;
NSString *NSRTFPropertyStackOverflowException;
NSString *NSAppKitIgnoredException;
NSString *NSBadComparisonException;
NSString *NSImageCacheException;
NSString *NSNibLoadingException;
NSString *NSBrowserIllegalDelegateException;
NSString *NSAccessibilityException;
```

Constants

`NSTextLineTooLongException`

Exception generated if a line is too long in a `NSText` object.

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSTextNoSelectionException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSWordTablesWriteException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSWordTablesReadException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSTextReadException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSTextWriteException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSPasteboardCommunicationException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSPrintingCommunicationException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSAbortModalException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSAbortPrintingException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSIllegalSelectorException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSAppKitVirtualMemoryException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSBadRTFDirectiveException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSBadRTFFontTableException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSBadRTFStyleSheetException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSTypedStreamVersionException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSTIFFException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSPrintPackageException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSBadRTFColorTableException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSDraggingException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSColorListIOException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSColorListNotEditableException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSBadBitmapParametersException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSWindowServerCommunicationException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSFontUnavailableException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSPPDIncludeNotFoundException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSPPDParseException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSPPDIncludeStackOverflowException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSPPDIncludeStackUnderflowException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSRTFPropertyStackOverflowException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSAppKitIgnoredException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSBadComparisonException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSImageCacheException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSNibLoadingException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSBrowserIllegalDelegateException`

Available in Mac OS X v10.0 and later.

Declared in `NSErrors.h`.

`NSAccessibilityException`

Available in Mac OS X v10.2 and later.

Declared in `NSErrors.h`.

Declared In
`NSErrors.h`

Document Revision History

This table describes the changes to *Application Kit Framework Reference*.

Date	Notes
2009-08-28	Added NSTextInputContext Class Reference and updated the class hierarchy diagrams.
2009-05-28	Added new classes introduced in Mac OS X v10.6.
2008-11-19	Added NSTextInputClient Protocol Reference.
2007-10-31	Updated framework illustrations.
2007-05-10	Added new classes introduced with Mac OS X v10.5.
2006-05-23	First publication of this content as a collection of separate documents.

Document Revision History