NSWindow

Inherits From: NSResponder : NSObject

Conforms To: NSCoding (NSResponder)
NSObject (NSObject)

Declared In: AppKit/NSWindow.h

Class Description

The NSWindow class defines objects that manage and coordinate the windows that an application displays on the screen. A single NSWindow object corresponds to, at most, one window. The two principle functions of an NSWindow are to provide an area in which views can be placed, and to accept and distribute, to the appropriate NSViews, events that the user instigates by manipulating the mouse and keyboard.

Rectangles, Views, and the View Hierarchy

An NSWindow is defined by a *frame rectangle* that encloses the entire window, including its title bar, resize bar, and border, and by a *content rectangle* that encloses just its content area. Both rectangles are specified in the screen coordinate system. The frame rectangle establishes the NSWindow's *base coordinate system*. This coordinate system is always aligned with and is measured in the same increments as the screen coordinate system (in other words, the base coordinate system can't be rotated or scaled). The origin of a base coordinate system is the bottom left corner of the window's frame rectangle.

You create an NSWindow (through one of the **init:...** methods) by specifying, among other attributes, the size and location of its content rectangle. The frame rectangle is derived from the dimensions of the content rectangle.

When it's created, an NSWindow automatically creates two NSViews: an opaque *frame view* and a transparent *content view* that fills the content area. The frame view is a private object that your application can't access directly. The content view is the ªhighestº accessible view

in the window; you can replace the content view with an NSView of your own creation through NSWindow's **setContentView:** method.

You add other views to the window by declaring each to be a subview of the content view, or a subview of one of the content view's subviews, and so on, through NSView's **addSubview:** method. This tree of views is called the window's *view hierarchy*. When an NSWindow is told to display itself, it does so by sending view-displaying messages to each object in its view hierarchy. Because displaying is carried out in a determined order, the content view (which is drawn first) may be wholly or partially obscured by its subviews, and these subviews may be obscured by their subviews (and so on).

Event Handling

The window system and the NSApplication object forward mouse and keyboard events to the appropriate NSWindow object. The NSWindow that's currently designated to receive keyboard events is known as the *key window*. If the mouse or keyboard event affects the window directlyÐresizing or moving it, for exampleÐthe NSWindow performs the appropriate operation itself and sends messages to its delegate informing it of its intentions, thus allowing your application to intercede. Events that are directed at specific views within the window are forwarded by the NSWindow to the NSView.

The NSWindow keeps track of the object that was last selected to handle keyboard events as its *first responder*. The first responder is typically the NSView that displays the current selection. In addition to keyboard events, the first responder is sent action messages that have a user-selected target (a **nil** target in program code). The NSWindow continually updates the first responder in response to the user's mouse actions.

Each NSWindow provides a *field editor*, an NSText object that handles small-scale text-editing chores. The field editor can be used by the NSWindow's first responder to edit the text that it displays. The **fieldEditor:forObject:** method returns the NSWindow's field editor. (You can make this method instead return an alternative NSText object, appropriate for the object specified the second argument, by implementing the delegate method **windowWillReturnFieldEditor:toObject:**.)

Initializing and Getting a New NSWindow Object

- (id)**initWithContentRect:**(NSRect)*contentRect* **styleMask:**(unsigned int)*aStyle* **backing:**(NSBackingStoreType)*bufferingType* **defer:**(BOOL)*flag*	Initializes the new window object with a location and size for content of *contentRect*, a window style and buttons as indicated in the bitmap mask *aStyle*, drawing buffering as indicated by *bufferingType*. If *flag* is YES, the window system defers creating the window until it's needed.
- (id)**initWithContentRect:**(NSRect)*contentRect*	Initializes the new window object for a screen as specified

styleMask:(unsigned int)*aStyle*
backing:(NSBackingStoreType)*bufferingType*
defer:(BOOL)*flag*
screen:(NSScreen *)*aScreen*

by *aScreen*, with a location and size for content of *contentRect*, a window style and buttons as indicated in the bitmap mask *aStyle*, drawing buffering as indicated by *bufferingType*. If *flag* is YES,the window system defers creating the window until it's needed.

Computing Frame and Content Rectangles

+ (NSRect)**contentRectForFrameRect:**(NSRect)*aRect*
styleMask:(unsigned int)*aStyle*

Gets the content rectangle for frame rectangle *aRect* in a window of type *aStyle*.

+ (NSRect)**frameRectForContentRect:**(NSRect)*aRect*
styleMask:(unsigned int)*aStyle*

Gets the frame rectangle for content rectangle *aRect* in a window of type *aStyle*.

+ (float)**minFrameWidthWithTitle:**(NSString *)*aTitle*
styleMask:(unsigned int)*aStyle*

Returns the minimum frame width needed for *aTitle* in a window of type *aStyle*.

Accessing the Content View

- (id)**contentView**

Returns the NSWindow's content view.

- (void)**setContentView:**(NSView *)*aView*

Makes *aView* the NSWindow's content view.

Window Graphics

- (NSColor *)**backgroundColor**

Returns the window's background color.

- (NSString *)**representedFilename**

Returns the filename associated with this window (regardless of the title string).

- (void)**setBackgroundColor:**(NSColor *)*color*

Sets the window's background color to *color*.

- (void)**setRepresentedFilename:**(NSString *)*aString*

Alters *aString* by formatting it as a path and filename, then sets the filename associated with this window to the result. If *filename* doesn't include a path to

the file, the current working directory is used. This method doesn't affect the title string.

- (void)**setTitle:**(NSString *)*aString* — Makes *aString* the window's title.

- (void)**setTitleWithRepresentedFilename:**(NSString *)*aString*
  Invokes **setRepresentedFilename:** and makes the resultant string the window's title.

- (unsigned int)**styleMask** — Returns the window's border and title-bar style.

- (NSString *)**title** — Returns the window's title string.

Window Device Attributes

- (NSBackingStoreType)**backingType** — Returns the type of the window device's backing store.

- (NSDictionary *)**deviceDescription** — Returns the window device's attributes as key/value pairs.

- (int)**gState** — Returns the graphics-state object for the window object.

- (BOOL)**isOneShot** — Returns whether backing-store memory for the window is freed when the window is ordered off-screen.

- (void)**setBackingType:**(NSBackingStoreType)*type* — Sets the type of window-device backing store.

- (void)**setOneShot:**(BOOL)*flag* — Sets whether backing-store memory for the window should be freed when the window is ordered off-screen.

- (int)**windowNumber** — Returns the window number.

The Miniwindow

- (NSImage *)**miniwindowImage** — Returns the image that's displayed in the miniwindow.

- (NSString *)**miniwindowTitle** — Returns the title that's displayed in the miniwindow.

- (void)**setMiniwindowImage:**(NSImage *)*image* — Sets the *image* that's displayed in the miniwindow.

- (void)**setMiniwindowTitle:**(NSString *)*title* — Sets the *title* that's displayed in the miniwindow.

The Field Editor

- (void)**endEditingFor:**(id)*anObject* — Ends the field editor's editing assignment for *anObject*.

- (NSText *)**fieldEditor:**(BOOL)*createFlag* **forObject:**(id)*anObject* — Returns the window object's field editor for *anObject*. If the field editor does not exist and *createFlag* is YES, creates a field editor.

Window Status and Ordering

- (void)**becomeKeyWindow** — Records the window's new status as the key window. This method posts the notification NSWindowDidBecomeKeyNotification with the receiving object to the default notification center.

- (void)**becomeMainWindow** — Records the window's new status as the main window. This method posts the notification NSWindowDidBecomeMainNotification with the receiving object to the default notification center.

- (BOOL)**canBecomeKeyWindow** — Returns whether the receiving window object can be the key window.

- (BOOL)**canBecomeMainWindow** — Returns whether the receiving window object can be the main window.

- (BOOL)**hidesOnDeactivate** — Returns whether deactivation hides the window.

- (BOOL)**isKeyWindow** — Returns whether the receiving window object is the key window.

- (BOOL)**isMainWindow** — Returns whether the receiving window object is the main window.

- (BOOL)**isMiniaturized** — Returns whether the window is hidden (and the miniwindow displayed).

- (BOOL)**isVisible** — Returns whether the window object is in the screen list (and thus visible).

- (int)**level** — Returns the current window level.

- (void)**makeKeyAndOrderFront:**(id)*sender* — Makes the receiving window object the key window and brings it forward.

- (void)**makeKeyWindow** — Makes the receiving window object the key window.

- (void)**makeMainWindow** — Makes the receiving window object the main window.

- (void)**orderBack:**(id)*sender* — Puts the window object at the back of its tier.

- (void)**orderFront:**(id)*sender* — Puts the window object at the front of its tier.

- (void)**orderFrontRegardless** — Puts the window object at the front even if the application is inactive. If the window is currently miniaturized, this method posts the notification NSWindowDidDeminiaturizeNotification with the window object to the default notification center.

- (void)**orderOut:**(id)*sender* — Removes the window object from the screen list.

- (void)**orderWindow:**(NSWindowOrderingMode)*place*
  relativeTo:(int)*otherWin* — Repositions the window object in the screen list in position *place* relative to another window. If the window is currently miniaturized, this method posts the NSWindowDidDeminiaturizeNotification notification with that window object to the default notification center.

- (void)**resignKeyWindow** — Records that the window object is no longer the key window. This method posts the notification NSWindowDidResignKeyNotification with the receiving object to the default notification center.

- (void)**resignMainWindow** — Records that the window object is no longer the main window. This method posts the notification NSWindowDidResignMainNotification with the receiving object to the default notification center.

- (void)**setHidesOnDeactivate:**(BOOL)*flag* — Sets whether deactivation hides the window.

- (void)**setLevel**:(int)*newLevel* — Resets the window level to *newLevel*.

Moving and Resizing the Window

- (NSPoint)**cascadeTopLeftFromPoint:**(NSPoint)*topLeftPoint*

When successively invoked, tiles windows by offsetting them slightly to the right and down from the previous window. Returns the top left point of the placed window, which is typically used for *topLeftPoint* in the next invocation. If you specify (0,0), places the window as is, and returns its top left point.

- (void)**center**

Centers the window on the screen.

- (NSRect)**constrainFrameRect:**(NSRect)*frameRect*
  toScreen:(NSScreen *)*screen*

Constrains the window's frame rectangle *frameRect* to *screen*. Returns the frame rectangle.

- (NSRect)**frame**

Returns the window's frame rectangle

- (NSSize)**minSize**

Returns the window's minimum size.

- (NSSize)**maxSize**

Returns the window's maximum size

- (void)**setContentSize:**(NSSize)*aSize*

Resizes the window's content area to *aSize.*

- (void)**setFrame:**(NSRect)*frameRect*
  display:(BOOL)*flag*

Moves and/or resizes the window frame to *frameRect*. *flag* determines whether the window is displayed. This method posts the NSWindowDidResizeNotification notification with the receiving object to the default notification center.

- (void)**setFrameOrigin:**(NSPoint)*aPoint*

Moves the window by changing its frame origin to *aPoint*.

- (void)**setFrameTopLeftPoint:**(NSPoint)*aPoint*

Moves the window by changing its top-left corner to *aPoint*.

- (void)**setMinSize:**(NSSize)*aSize*

Sets the window's minimum size.

- (void)**setMaxSize:**(NSSize)*aSize*

Sets the window's maximum size.

Converting Coordinates

- (NSPoint)**convertBaseToScreen:**(NSPoint)*aPoint*

Converts *aPoint* from base to screen coordinates.

- (NSPoint)**convertScreenToBase:**(NSPoint)*aPoint*

  Converts *aPoint* from screen to base coordinates.

Managing the Display

- (void)**display** — Displays all the window's views.
- (void)**disableFlushWindow** — Disables flushing for a buffered window.
- (void)**displayIfNeeded** — Displays all the window's views that need to be redrawn.
- (void)**enableFlushWindow** — Enables flushing for a buffered window.
- (void)**flushWindow** — Flushes the window's buffer to the screen.
- (void)**flushWindowIfNeeded** — Conditionally flushes the window's buffer to the screen.
- (BOOL)**isAutodisplay** — Returns whether the window displays all views requiring redrawing when **update** is invoked.
- (BOOL)**isFlushWindowDisabled** — Returns whether flushing is disabled.
- (void)**setAutodisplay**:(BOOL)*flag* — Sets whether the window displays all views requiring redrawing when **update** is invoked.
- (void)**setViewsNeedDisplay:**(BOOL)*flag* — Sets whether some views of the receiving window object should be redrawn.
- (void)**update** — Update's the window's display and cursor rectangles. This method is invoked after every event. When it successfully completes, it posts the NSWindowDidUpdateNotification notification.
- (void)**useOptimizedDrawing:**(BOOL)*flag* — Sets whether the window's views should optimize drawing.
- (BOOL)**viewsNeedDisplay** — Returns whether some views of the receiving NSWindow object should be redrawn.

Screens and Window Depths

+ (NSWindowDepth)**defaultDepthLimit**	Returns the default depth limit for all windows.
- (BOOL)**canStoreColor**	Returns whether the window is deep enough to store colors.
- (NSScreen *)**deepestScreen**	Returns the deepest screen that the window is on.
- (NSWindowDepth)**depthLimit**	Returns the window's depth limit.
- (BOOL)**hasDynamicDepthLimit**	Returns whether the depth limit depends on the screen.
- (NSScreen *)**screen**	Returns the screen that (most of) the window is on.
- (void)**setDepthLimit:**(NSWindowDepth)*limit*	Sets the window's depth limit to *limit*
- (void)**setDynamicDepthLimit:**(BOOL)*flag*	Sets whether the depth limit will depend on the screen.

Cursor Management

- (BOOL)**areCursorRectsEnabled**	Returns whether cursor rectangles are enabled.
- (void)**disableCursorRects**	Disables all cursor rectangles in the window object.
- (void)**discardCursorRects**	Removes all cursor rectangles in the window object.
- (void)**enableCursorRects**	Enables cursor rectangles in the window object.
- (void)**invalidateCursorRectsForView:**(NSView *)*aView*	Marks cursor rectangles invalid for *aView*.
- (void)**resetCursorRects**	Resets cursor rectangles for the window object.

Handling User Actions and Events

- (void)**close**	Closes the window. When this method begins, it posts the notification NSWindowWillCloseNotification with the receiving object to the default notification center.
- (void)**deminiaturize:**(id)*sender*	Hides the miniwindow and redisplays the window.

- (BOOL)**isDocumentEdited** Returns whether the window's document has been edited.

- (BOOL)**isReleasedWhenClosed** Returns whether the window object is released when it is closed.

- (void)**miniaturize:**(id)*sender* Hides the window and displays its miniwindow. When this method begins, it posts the notification NSWindowWillMiniaturizeNotification with the receiving object to the default notification center. When it completes successfully, it posts NSWindowDidMiniaturizeNotification.

- (void)**performClose:**(id)*sender* Simulates user clicking the close button.

- (void)**performMiniaturize:**(id)*sender* Simulates user clicking the miniaturize button.

- (int)**resizeFlags** Returns the event modifier flags during resizing.

- (void)**setDocumentEdited:**(BOOL)*flag* Sets whether the window's document has been edited.

- (void)**setReleasedWhenClosed:**(BOOL)*flag* Sets whether closing the window object also releases it.

Aiding Event Handling

- (BOOL)**acceptsMouseMovedEvents** Returns whether the NSWindow accepts mouse-moved events.

- *(*NSEvent *)**currentEvent** Returns the current event object for the application.

- (void)**discardEventsMatchingMask:**(unsigned int)*mask*
  beforeEvent:(NSEvent *)*lastEvent* Discards any events in the event queue that have a type indicated by bitmap *mask* until the method encounters the event *lastEvent*.

- (NSResponder *)**firstResponder** Returns the first responder to user events.

- (void)**keyDown:**(NSEvent *)*theEvent* Handles key-down events.

- (BOOL)**makeFirstResponder:**(NSResponder *)*aResponder*
  Makes *aResponder* the first responder to user events.

- (NSPoint)**mouseLocationOutsideOfEventStream** Provides current location of the cursor.

- (NSEvent *)**nextEventMatchingMask:**(unsigned int)*mask*

Returns the next event object for the application that matches the events indicated by event mask *mask*.

- (NSEvent *)**nextEventMatchingMask:**(unsigned int)*mask*
untilDate:(NSDate *)*expiration*
inMode:(NSString *)*mode*
dequeue:(BOOL)*deqFlag*

Returns the next event object for the application that matches the events indicated by event mask *mask*, and that occurs before time *expiration*; until *expiration*, the run loop runs in *mode*.

- (void)**postEvent:**(NSEvent *)*event*
atStart:(BOOL)*flag*

Post an *event* for the application; if *atStart* is YES, the event goes to the beginning of the event queue.

- (void)**setAcceptsMouseMovedEvents:**(BOOL)*flag*

Sets whether the NSWindow accepts mouse-moved events.

- (void)**sendEvent:**(NSEvent *)*theEvent*

Dispatches mouse and keyboard events. If this method is dispatching a window exposed event, it posts the NSWindowDidExposeNotification notification with the receiving object and, in the notification's dictionary, a rectangle describing the exposed area (with the key NSExposedRect) to the default notification center. If it is dispatching a screen changed event, it posts NSWindowDidChangeScreenNotification with the receiving object. If it is dispatching a window moved event, it posts NSWindowDidMoveNotification.

- (BOOL)**tryToPerform:**(SEL)*anAction*
with:(id)*anObject*

Aids in dispatching action messages (*anAction*) to *anObject*.

- (BOOL)**worksWhenModal**

Override to return whether the window object accepts events when a modal panel is being run. Default is NO.

Dragging

- (void)**dragImage:**(NSImage *)*anImage*
at:(NSPoint)*baseLocation*
offset:(NSSize)*initialOffset*

Initiates an image-dragging session. NSView invokes this method inside its implementation of **mouseDown:**.

event:(NSEvent *)*event*
pasteboard:(NSPasteboard *)*pboard*
source:(id)*sourceObject*
slideBack:(BOOL)*slideFlag*

- (void)**registerForDraggedTypes:**(NSArray *)*newTypes*
Registers the NSPasteboard types (*newTypes*) that the window object accepts in an image-dragging session.

- (void)**unregisterDraggedTypes** Unregisters the window object as a recipient of dragged images.

Services and Windows Menu Support

- (BOOL)**isExcludedFromWindowsMenu** Returns whether the receiving window object is omitted from the Windows menu.

- (void)**setExcludedFromWindowsMenu:**(BOOL)*flag*
Sets whether the receiving window object is omitted from the Windows menu.

- (id)**validRequestorForSendType:**(NSString *)*sendType*
returnType:(NSString *)*returnType* Returns whether the window can respond to a service with send and receive types *sendType* and *returnType*.

Saving and Restoring the Frame

+ (void)**removeFrameUsingName:**(NSString *)*name*
Removes the named frame rectangle from the system defaults.

- (NSString *)**frameAutosaveName** Returns the name that's used to autosave the frame rectangle as a system default.

- (void)**saveFrameUsingName:**(NSString *)*name* Saves the frame rectangle as a system default.

- (BOOL)**setFrameAutosaveName:**(NSString *)*name*
Sets the *name* that's used to autosave the frame rectangle as a system default.

- (void)**setFrameFromString:**(NSString *)*string* Sets the frame rectangle from *string*, which encodes the position and dimensions of

the frame rectangle and the position and dimensions of the screen.

- (BOOL)**setFrameUsingName:**(NSString *)*name* — Sets the frame rectangle from the named default.

- (NSString *)**stringWithSavedFrame** — Returns a string encoding the position and dimensions of the frame rectangle and the position and dimensions of the screen.

Printing and PostScript

- (NSData *)**dataWithEPSInsideRect:**(NSRect)*rect*

  Returns the encapsulated PostScript inside *rect* as a data object.

- (void)**fax:**(id)*sender* — Faxes all the window's views.

- (void)**print:**(id)*sender* — Prints all the window's views.

Assigning a Delegate

- (id)**delegate** — Returns the window object's delegate.

- (void)**setDelegate:**(id)*anObject* — Makes *anObject* the window object's delegate.

Implemented by the Delegate

- (BOOL)**windowShouldClose:**(id)*sender* — Notifies delegate that the window is about to close.

- (NSSize)**windowWillResize:**(NSWindow *)*sender*
  toSize:(NSSize)*frameSize* — Lets delegate constrain resizing to *frameSize*.

- (id)**windowWillReturnFieldEditor:**(NSWindow *)*sender*
  toObject:(id)*client* — Lets delegate provide another text object for field editor.

- (void)**windowDidBecomeKey:**(NSNotification *)*aNotification*

  Sent by the default notification center to notify the delegate that the window is the key window. *aNotification* is always NSWindowDidBecomeKeyNotification. If

the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowDidBecomeMain:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window is the main window. *aNotification* is always NSWindowDidBecomeMainNotification. If the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowDidChangeScreen:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window changed screens. *aNotification* is always NSWindowDidChangeScreenNotification. If the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowDidDeminiaturize:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window was restored to screen. *aNotification* is always NSWindowDidDeminiaturizeNotification. If the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowDidExpose:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window was exposed. *aNotification* is always NSWindowDidExposeNotification. If the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowDidMiniaturize:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window was miniaturized. *aNotification* is always NSWindowDidMiniaturizeNotification. If the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowDidMove:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window did

move. *aNotification* is always NSWindowDidMoveNotification. If the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowDidResignKey:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window isn't the key window. *aNotification* is always NSWindowDidResignKeyNotification. If the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowDidResignMain:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window isn't the main window. *aNotification* is always NSWindowDidResignMainNotification. If the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowDidResize:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window was resized. *aNotification* is always NSWindowDidResizeNotification. If the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowDidUpdate:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window was updated. *aNotification* is always NSWindowDidUpdateNotification. If the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowWillClose:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window will close. *aNotification* is always NSWindowWillCloseNotification. If the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowWillMiniaturize:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window will be miniaturized. *aNotification* is always NSWindowWillMiniaturizeNotification. If

the delegate implements this method, it's automatically registered to receive this notification.

- (void)**windowWillMove:**(NSNotification *)*aNotification*

Sent by the default notification center to notify the delegate that the window will move. *aNotification* is always NSWindowWillMoveNotification. If the delegate implements this method, it's automatically registered to receive this notification.