Types and Constants

Application

id **NSApp**;

Represents the application's NSApplication object.

typedef struct _NSModalSession ***NSModalSession**;

This structure stores information used by the system during a modal session.

enum {
    NSRunStoppedResponse,
    NSRunAbortedResponse,
    NSRunContinuesResponse
};

Predefined return values for **runModalFor:** and **runModalSession:**.

NSString ***NSModalPanelRunLoopMode**;

NSString ***NSEventTrackingRunLoopMode**;

Input-filter modes passed to NSRunLoop.

Box

typedef enum _NSTitlePosition {
    NSNoTitle,
    NSAboveTop,
    NSAtTop,
    NSBelowTop,
    NSAboveBottom,
    NSAtBottom,

This type's constants represent the locations where an NSBox's title is placed in relation to the border (**setTitlePosition:** and **titlePosition**).

NSBelowBottom
} **NSTitlePosition**;

Buttons

typedef enum _NSButtonType {
NSMomentaryPushButton,
NSPushOnPushOffButton,
NSToggleButton,
NSSwitchButton,
NSRadioButton,
NSMomentaryChangeButton,
NSOnOffButton
} **NSButtonType**;

The constants of **NSButtonType** indicate the way NSButtons and NSButtonCells behave when pressed, and how they display their state. They are used in NSButton's **setType:** method.

Cells and Button Cells

typedef enum _NSCellType {
NSNullCellType,
NSTextCellType,
NSImageCellType
} **NSCellType**;

Represent different types of NSCell objects.
No display.
Displays text.
Displays an image.
Returned from **type** and set via **setType:**.

typedef enum _NSCellImagePosition {
NSNoImage,
NSImageOnly,
NSImageLeft,
NSImageRight,
NSImageBelow,
NSImageAbove,
NSImageOverlaps
} **NSCellImagePosition**;

Represent the position of an NSButtonCell relative to its title. Returned from **imagePosition** and set through **setImagePosition:**.

typedef enum _NSCellAttribute {
    NSCellDisabled,
    NSCellState,
    NSPushInCell,
    NSCellEditable,
    NSChangeGrayCell,
    NSCellHighlighted,
    NSCellLightsByContents,
    NSCellLightsByGray,
    NSChangeBackgroundCell,
    NSCellLightsByBackground,
    NSCellIsBordered,
    NSCellHasOverlappingImage,
    NSCellHasImageHorizontal,
    NSCellHasImageOnLeftOrBottom,
    NSCellChangesContents,
    NSCellIsInsetButton
} **NSCellAttribute**;

The constant values of **NSCellAttribute** represent parameters that you can set and access through NSCell's and NSButtonCell's **setParameter:to:** and **getParameter:** methods. Only the first five constants are used by NSCell; the others apply to NSButtonCells only.

enum {
    NSAnyType,
    NSIntType,
    NSPositiveIntType,
    NSFloatType,
    NSPositiveFloatType,
    NSDateType,
    NSDoubleType,
    NSPositiveDoubleType
};

Numeric data types that an NSCell can accept. Used as the argument for **setEntryType:**.

enum {
    NSNoCellMask,
    NSContentsCellMask,
    NSPushInCellMask,
    NSChangeGrayCellMask,
    NSChangeBackgroundCellMask

NSButtonCell uses these values to determine how to highlight a button cell or show an ON state (returned/passed in **showsStateBy**/**setShowsStateBy** and **highlightsBy**/**setHighlightsBy**).

};

Color

enum {
    NSGrayModeColorPanel,
    NSRGBModeColorPanel,
    NSCMYKModeColorPanel,
    NSHSBModeColorPanel,
    NSCustomPaletteModeColorPanel,
    NSColorListModeColorPanel,
    NSWheelModeColorPanel
};

Tags that identify modes (or views) in the color panel.

enum {
    NSColorPanelGrayModeMask,
    NSColorPanelRGBModeMask,
    NSColorPanelCMYKModeMask,
    NSColorPanelHSBModeMask,
    NSColorPanelCustomPaletteModeMask,
    NSColorPanelColorListModeMask,
    NSColorPanelWheelModeMask,
    NSColorPanelAllModesMask
};

Bit masks for determining the current mode (or view) of the color panel.

Data Link

typedef int **NSDataLinkNumber**;

Returned by NSDataLink's **linkNumber** method as a persistent identifier of a destination link.

NSString ***NSDataLinkFileNameExtension**;

The file name suffix to be used when data links are saved. The default is **objlink**.

typedef enum _NSDataLinkDisposition {
    NSLinkInDestination,

Returned by NSDataLink's **disposition** method to identify a link as a destination link, a source link, or a broken

NSLinkInSource,
NSLinkBroken
} **NSDataLinkDisposition**;

link.

typedef enum _NSDataLinkUpdateMode {
NSUpdateContinuously,
NSUpdateWhenSourceSaved,
NSUpdateManually,
NSUpdateNever
} **NSDataLinkUpdateMode**;

Identifies when a link's data is to be updated. Set through the **setUpdateMode:** method and returned by **updateMode**.

Drag Operation

typedef enum _NSDragOperation {
NSDragOperationNone,
NSDragOperationCopy,
NSDragOperationLink,
NSDragOperationGeneric,
NSDragOperationPrivate,
NSDragOperationAll
} **NSDragOperation**;

The constants of this type identify different kinds of dragging operations. **NSDragOperationNone** implies that the operation is rejected. **NSDragOperationPrivate** means that the system leaves the cursor alone.

Event Handling

typedef enum _NSEventType {
NSLeftMouseDown,
NSLeftMouseUp,
NSRightMouseDown,
NSRightMouseUp,
NSMouseMoved,
NSLeftMouseDragged,
NSRightMouseDragged,

Each constant of **NSEventType** identifies an event type. (See the NSEvent class description.)

```
        NSMouseEntered,
        NSMouseExited,
        NSKeyDown,
        NSKeyUp,
        NSFlagsChanged,
        NSPeriodic,
        NSCursorUpdate
} NSEventType;
```

```
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,
```

Unicodes that identify function keys on the keyboard, OpenStep reserves the range 0xF700-0xF8FF for this purpose. The availability of some keys is system-dependent.

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
};

enum {
NSAlphaShiftKeyMask,
NSShiftKeyMask,
NSControlKeyMask,
NSAlternateKeyMask,
NSCommandKeyMask,
NSNumericPadKeyMask,
NSHelpKeyMask,
NSFunctionKeyMask
};

Device-independent bit masks for evaluating event-modifier flags to determine which modifier key (if any) was pressed.

enum {
NSLeftMouseDownMask,
NSLeftMouseUpMask,
NSRightMouseDownMask,
NSRightMouseUpMask,
NSMouseMovedMask,
NSLeftMouseDraggedMask,
NSRightMouseDraggedMask,
NSMouseEnteredMask,
NSMouseExitedMask,
NSKeyDownMask,
NSKeyUpMask,
NSFlagsChangedMask,

Bit masks for determining the type of events.

NSPeriodicMask,
NSCursorUpdateMask,
NSAnyEventMask
};

Exceptions

Global Exception Strings

The following global strings identify the exceptions returned by various operations in the Application Kit. They are defined in NSErrors.h.

NSString ***NSAbortModalException**;

NSString ***NSAbortPrintingException**;

NSString ***NSAppKitIgnoredException**;

NSString ***NSAppKitVirtualMemoryException**;

NSString ***NSBadBitmapParametersException**;

NSString ***NSBadComparisonException**;

NSString ***NSBadRTFColorTableException**;

NSString ***NSBadRTFDirectiveException**;

NSString ***NSBadRTFFontTableException**;

NSString ***NSBadRTFStyleSheetException**;

NSString ***NSBrowserIllegalDelegateException**;

NSString ***NSColorListIOException**;

NSString ***NSColorListNotEditableException**;

NSString ***NSDraggingException**;

NSString ***NSFontUnavailableException**;

NSString ***NSIllegalSelectorException**;

NSString ***NSImageCacheException**;

NSString ***NSNibLoadingException**;

NSString ***NSPPDIncludeNotFoundException**;

NSString ***NSPPDIncludeStackOverflowException**;

NSString ***NSPPDIncludeStackUnderflowException**;

NSString ***NSPPDParseException**;

NSString ***NSPasteboardCommunicationException**;

NSString ***NSPrintOperationExistsException**; (Defined in NSPrintOperation.h.)

NSString ***NSPrintPackageException**;

NSString ***NSPrintingCommunicationException**;

NSString ***NSRTFPropertyStackOverflowException**;

NSString ***NSTIFFException**;

NSString ***NSTextLineTooLongException**;

NSString ***NSTextNoSelectionException**;

NSString ***NSTextReadException**;

NSString ***NSTextWriteException**;

NSString ***NSTypedStreamVersionException**;

NSString ***NSWindowServerCommunicationException**;

NSString ***NSWordTablesReadException**;

NSString ***NSWordTablesWriteException**;

Fonts

typedef unsigned int **NSFontTraitMask**;

Characterizes one or more of a font's traits. It's used as an argument type for several of the methods in the NSFontManager class. You build a mask by OR'ing together the following enumeration constants.

enum {
    NSItalicFontMask,
    NSBoldFontMask,
    NSUnboldFontMask,
    NSNonStandardCharacterSetFontMask,
    NSNarrowFontMask,
    NSExpandedFontMask,
    NSCondensedFontMask,
    NSSmallCapsFontMask,
    NSPosterFontMask,
    NSCompressedFontMask,
    NSUnitalicFontMask
};

Values used by NSFontManager to identify font traits.

typedef unsigned int **NSGlyph**;

A type for numbers identifying font glyphs. It's used as the argument type for several of the methods in NSFont.

enum {
    NSFPPreviewButton ,
    NSFPRevertButton,
    NSFPSetButton,
    NSFPPreviewField,
    NSFPSizeField,
    NSFPSizeTitle,
    NSFPCurrentField
};

Tags identifying views in the font panel.

const float ***NSFontIdentityMatrix**;

Identifies a font matrix that's used for fonts displayed in an NSView object that has an unflipped coordinate system.

NSString ***NSAFMAscender**;

Global keys to access the values available in the AFM

NSString ***NSAFMCapHeight**;
NSString ***NSAFMCharacterSet**;
NSString ***NSAFMDescender**;
NSString ***NSAFMEncodingScheme**;
NSString ***NSAFMFamilyName**;
NSString ***NSAFMFontName**;
NSString ***NSAFMFormatVersion**;
NSString ***NSAFMFullName**;
NSString ***NSAFMItalicAngle**;
NSString ***NSAFMMappingScheme**;
NSString ***NSAFMNotice**;
NSString ***NSAFMUnderlinePosition**;
NSString ***NSAFMUnderlineThickness**;
NSString ***NSAFMVersion**;
NSString ***NSAFMWeight**;
NSString ***NSAFMXHeight**;

dictionary. You can convert the appropriate values (e.g., ascender, cap height) to floating point values by using NSString's **floatValue** method.

Graphics

typedef int **NSWindowDepth**

This type gives the window-depth limit. Use the NSAvailableWindowDepths() function to get a list of available window depths. Use the functions **NSBitsPerSampleFromDepth()**, **NSBitsPerPixelFromDepth()**, **NSPlanarFromDepth**, and **NSColorSpaceFromDepth()** to extract information from a window depth. The NSWindowDepth type is also used as an argument type of methods in NSScreen and NSWindow.

typedef enum _NSTIFFCompression {
    NSTIFFCompressionNone = 1,
    NSTIFFCompressionCCITTFAX3 = 3,
    NSTIFFCompressionCCITTFAX4 = 4,
    NSTIFFCompressionLZW = 5,
    NSTIFFCompressionJPEG = 6,
    NSTIFFCompressionNEXT = 32766,
    NSTIFFCompressionPackBits = 32773,
    NSTIFFCompressionOldJPEG = 32865

The constants defined in this type represent the various TIFF (*tag image file format*) data compression schemes. They are defined in NSBitMapImageRep and used in several methods of that class as well as in the **TIFFRepresentationUsingCompression:factor:** method of NSImage.

} **NSTIFFCompression**;

enum {
    NSImageRepMatchesDevice
};

NSImageRepMatchesDevice indicates that the value varies according to the output device. It can be passed in (or received back) as the value of NSImageRep's **bitsPerSample**, **pixelsWide**, and **pixelsHigh**.

Colorspace Names

Predefined colorspace names. Used as arguments in **NSDrawBitMap()** and **NSNumberOfColorComponents()**; value returned from **NSColorSpaceFromDepth()**.

NSString ***NSCalibratedWhiteColorSpace**;

NSString ***NSCalibratedBlackColorSpace**;

NSString ***NSCalibratedRGBColorSpace**;

NSString ***NSDeviceWhiteColorSpace**;

NSString ***NSDeviceBlackColorSpace**;

NSString ***NSDeviceRGBColorSpace**;

NSString ***NSDeviceCMYKColorSpace**;

NSString ***NSNamedColorSpace**;

NSString ***NSCustomColorSpace**;

Gray Values

Standard gray values for the 2-bit deep grayscale colorspace.

const float **NSBlack**;

const float **NSDarkGray**;

const float **NSWhite**;

const float **NSLightGray**;

Device Dictionary Keys

Keys to get designated values from device dictionaries.

NSString ***NSDeviceResolution**;

NSString ***NSDeviceColorSpaceName**

NSString ***NSDeviceBitsPerSample**;

NSString ***NSDeviceIsScreen**;

NSString ***NSDeviceIsPrinter**;

NSString ***NSDeviceSize**;

Matrix

typedef enum _NSMatrixMode {
    NSRadioModeMatrix,
    NSHighlightModeMatrix,
    NSListModeMatrix,
    NSTrackModeMatrix
} **NSMatrixMode**;

The constants in this type represent the modes of operation of an NSMatrix.

Notifications

Notifications are posted to all interested observers of a specific condition to alert them that the condition has occurred. Global strings contain the actual text of the notification. In the Application Kit, these are defined per class. See the Foundation's NSNotification and NSNotificationCenter for details.

NSString ***NSApplicationDidBecomeActiveNotification**; NSApplication

NSString ***NSApplicationDidFinishLaunchingNotification**;

NSString ***NSApplicationDidHideNotification**;

NSString ***NSApplicationDidResignActiveNotification**;

NSString ***NSApplicationDidUnhideNotification**;

NSString ***NSApplicationDidUpdateNotification**;

NSString ***NSApplicationWillBecomeActiveNotification**;

NSString ***NSApplicationWillFinishLaunchingNotification**;

NSString ***NSApplicationWillHideNotification**;

NSString ***NSApplicationWillResignActiveNotification**;

NSString ***NSApplicationWillUnhideNotification**;

NSString ***NSApplicationWillUpdateNotification**;

NSString ***NSColorListChangedNotification**; NSColorList

NSString ***NSColorPanelColorChangedNotification**; NSColorPanel

NSString ***NSControlTextDidBeginEditingNotification**;NSControl

NSString ***NSControlTextDidEndEditingNotification**;

NSString ***NSControlTextDidChangeNotification**;

NSString ***NSImageRepRegistryChangedNotification**; NSImageRep

NSString ***NSSplitViewDidResizeSubviewsNotification**; NSSplitView

NSString ***NSSplitViewWillResizeSubviewsNotification**;

NSString ***NSTextDidBeginEditingNotification;** NSText

NSString ***NSTextDidEndEditingNotification;**

NSString ***NSTextDidChangeNotification;**

NSString ***NSViewFrameChangedNotification**; NSView

NSString ***NSViewFocusChangedNotification**;

NSString ***NSWindowDidBecomeKeyNotification**; NSWindow

NSString ***NSWindowDidBecomeMainNotification**;

NSString ***NSWindowDidChangeScreenNotification**;

NSString ***NSWindowDidDeminiaturizeNotification**;

NSString ***NSWindowDidExposeNotification**;

NSString ***NSWindowDidMiniaturizeNotification**;

NSString ***NSWindowDidMoveNotification**;

NSString ***NSWindowDidResignKeyNotification**;

NSString ***NSWindowDidResignMainNotification**;

NSString ***NSWindowDidResizeNotification**;

NSString ***NSWindowDidUpdateNotification**;

NSString ***NSWindowWillCloseNotification**;

NSString ***NSWindowWillMiniaturizeNotification**;

NSString ***NSWindowWillMoveNotification**;

NSString ***NSWorkspaceDidLaunchApplicationNotification**; NSWorkspace

NSString ***NSWorkspaceDidMountNotification**;

NSString ***NSWorkspaceDidPerformFileOperationNotification**;

NSString ***NSWorkspaceDidTerminateApplicationNotification**;

NSString ***NSWorkspaceDidUnmountNotification**;

NSString ***NSWorkspaceWillLaunchApplicationNotification**;

NSString ***NSWorkspaceWillPowerOffNotification**;

NSString ***NSWorkspaceWillUnmountNotification**;

Panel

```
enum {
        NSOKButton = 1,
        NSCancelButton = 0
};
```

Values returned by the standard panel buttons, OK and Cancel.

```
enum {
        NSAlertDefaultReturn = 1,
        NSAlertAlternateReturn = 0,
        NSAlertOtherReturn = -1,
        NSAlertErrorReturn   = -2
};
```

Values returned by the **NSRunAlertPanel()** function and by **runModalSession:** when the modal session is run with a Panel provided by **NSGetAlertPanel()**.

Page Layout

```
enum {
        NSPLImageButton,
        NSPLTitleField,
        NSPLPaperNameButton,
```

Tags that identify buttons, fields, and other views of the Page Layout panel.

NSPLUnitsButton,
NSPLWidthForm,
NSPLHeightForm,
NSPLOrientationMatrix,
NSPLCancelButton,
NSPLOKButton
};

Pasteboard

Pasteboard Type Globals

Identifies the standard pasteboard types. These are used in a variety of NSPasteboard methods and functions.

NSString ***NSStringPboardType**;

NSString ***NSColorPboardType**;

NSString ***NSFileContentsPboardType**;

NSString ***NSFilenamesPboardType**;

NSString ***NSFontPboardType**;

NSString ***NSRulerPboardType**;

NSString ***NSPostScriptPboardType**;

NSString ***NSTabularTextPboardType**;

NSString ***NSRTFPboardType**;

NSString ***NSTIFFPboardType**;

NSString ***NSDataLinkPboardType**; (Defined in NSDataLink.h.)

NSString ***NSGeneralPboardType**; (Defined in NSSelection.h.)

Pasteboard Name Globals

Identifies the standard pasteboard names. Used in class method **pasteboardWithName:** to get a pasteboard by name.

NSString ***NSDragPboard**;

NSString ***NSFindPboard**;

NSString ***NSFontPboard**;

NSString ***NSGeneralPboard**;

NSString ***NSRulerPboard**;

Printing

typedef enum _NSPrinterTableStatus {
    NSPrinterTableOK,
    NSPrinterTableNotFound,
    NSPrinterTableError
} **NSPrinterTableStatus**;

These constants describe the state of a printer-information table stored by an NSPrinter object. It is the argument type of the return value of **statusForTable:**.

typedef enum _NSPrintingOrientation {
    NSPortraitOrientation,
    NSLandscapeOrientation
} **NSPrintingOrientation**;

These constants represent the way a page is oriented for printing.

typedef enum _NSPrintingPageOrder {
    NSDescendingPageOrder,
    NSSpecialPageOrder,
    NSAscendingPageOrder,
    NSUnknownPageOrder
} **NSPrintingPageOrder**;

These constants describe the order in which pages are spooled for printing. **NSSpecialPageOrder** tells the spooler not to rearrange pages. Set through NSPrintingOperation's **setPageOrder:** method and returned by its **pageOrder** method.

typedef enum _NSPrintingPaginationMode {
    NSAutoPagination,
    NSFitPagination,
    NSClipPagination

These constants represent the different ways an image is divided into pages during pagination. Pagination can occur automatically, the image can be forced onto a page, or it can be clipped to a page.

} **NSPrintingPaginationMode**;

enum {
    NSPPSaveButton,
    NSPPPreviewButton,
    NSFaxButton,
    NSPPTitleField,
    NSPPImageButton,
    NSPPNameTitle,
    NSPPNameField,
    NSPPNoteTitle,
    NSPPNoteField,
    NSPPStatusTitle,
    NSPPStatusField,
    NSPPCopiesField,
    NSPPPageChoiceMatrix,
    NSPPPageRangeFrom,
    NSPPPageRangeTo,
    NSPPScaleField,
    NSPPOptionsButton,
    NSPPPaperFeedButton,
    NSPPLayoutButton
};

Tags that identify text fields, controls, and other views in the Print Panel.

Printing Information Dictionary Keys

The keys in the mutable dictionary associated with NSPrintingInfo. See NSPrintingInfo.h for types and descriptions of values.

NSString ***NSPrintAllPages**;

NSString ***NSPrintBottomMargin**;

NSString ***NSPrintCopies**;

NSString ***NSPrintFaxCoverSheetName**;

NSString ***NSPrintFaxHighResolution**;

NSString ***NSPrintFaxModem**;

NSString ***NSPrintFaxReceiverNames**;

NSString ***NSPrintFaxReceiverNumbers**;

NSString ***NSPrintFaxReturnReceipt**;

NSString ***NSPrintFaxSendTime**;

NSString ***NSPrintFaxTrimPageEnds**;

NSString ***NSPrintFaxUseCoverSheet**;

NSString ***NSPrintFirstPage**;

NSString ***NSPrintHorizonalPagination**;

NSString ***NSPrintHorizontallyCentered**;

NSString ***NSPrintJobDisposition**;

NSString ***NSPrintJobFeatures**;

NSString ***NSPrintLastPage**;

NSString ***NSPrintLeftMargin**;

NSString ***NSPrintManualFeed**;

NSString ***NSPrintOrientation**;

NSString ***NSPrintPackageException**;

NSString ***NSPrintPagesPerSheet**;

NSString ***NSPrintPaperFeed**;

NSString ***NSPrintPaperName**;

NSString ***NSPrintPaperSize**;

NSString ***NSPrintPrinter**;

NSString ***NSPrintReversePageOrder**;

NSString ***NSPrintRightMargin**;

NSString ***NSPrintSavePath**;

NSString ***NSPrintScalingFactor**;

NSString ***NSPrintTopMargin**;

NSString ***NSPrintVerticalPagination**;

NSString ***NSPrintVerticallyCentered**;

Print Job Disposition Values

These global constants define the disposition of a print job. See NSPrintInfo's **setJobDisposition:** and **jobDisposition**.

NSString ***NSPrintCancelJob**;

NSString ***NSPrintFaxJob**;

NSString ***NSPrintPreviewJob**;

NSString ***NSPrintSaveJob**;

NSString ***NSPrintSpoolJob**;

Save Panel

enum {
NSFileHandlingPanelImageButton,
NSFileHandlingPanelTitleField,
NSFileHandlingPanelBrowser,
NSFileHandlingPanelCancelButton,
NSFileHandlingPanelOKButton,
NSFileHandlingPanelForm,
NSFileHandlingPanelHomeButton,
NSFileHandlingPanelDiskButton,
NSFileHandlingPanelDiskEjectButton

Tags that identify buttons, fields, and other views in the Save Panel.

};

Scroller

typedef enum _NSScrollArrowPosition {
    NSScrollerArrowsMaxEnd,
    NSScrollerArrowsMinEnd,
    NSScrollerArrowsNone
} **NSScrollArrowPosition**;

NSScroller uses these constants in its **setArrowPosition:** method to set the position of the arrows within the scroller.

typedef enum _NSScrollerPart {
    NSScrollerNoPart,
    NSScrollerDecrementPage,
    NSScrollerKnob,
    NSScrollerIncrementPage,
    NSScrollerDecrementLine,
    NSScrollerIncrementLine,
    NSScrollerKnobSlot
} **NSScrollerPart**;

NSScroller uses these constants in its **hitPart** method to identify the part of the scroller specified in a mouse event.

typedef enum _NSScrollerUsablePart {
    NSNoScrollerParts,
    NSOnlyScrollerArrows,
    NSAllScrollerParts
} **NSUsableScrollerParts**;

These constants define the usable parts of an NSScroller object.

typedef enum _NSScrollerArrow {
    NSScrollerIncrementArrow,
    NSScrollerDecrementArrow
} **NSScrollerArrow**;

These constants indicate the two types of scroller arrow. NSScroller's **drawArrow:highlight:** method takes an NSScrollerArrow as the first argument.

const float **NSScrollerWidth**;

Identifies the default width of a vertical NSScroller object and the default height of a horizontal NSScroller object.

```
typedef struct _NSBreakArray {
        NSTextChunk chunk;
        NSLineDesc   breaks[1];
} NSBreakArray;
```

Holds line-break information for an NSText object. It's mainly an array of line descriptors.

```
typedef struct _NSCharArray {
        NSTextChunk chunk;
        unsigned char text[1];
} NSCharArray;
```

Holds the character array for the current line in the NSText object.

```
typedef unsigned short (*NSCharFilterFunc) (
        unsigned short charCode,
        int flags,
        NSStringEncoding theEncoding);
```

The character filter function analyzes each character the user enters in the NSText object.

```
typedef struct _NSFSM {
        const struct _NSFSM   *next;
        short     delta;
        short     token;
} NSFSM;
```

A word definition finite-state machine structure used by an NSText object.

```
typedef struct _NSHeightChange {
        NSLineDesc   lineDesc;
        NSHeightInfo heightInfo;
} NSHeightChange;
```

Associates line descriptors and line-height information in an NSText object.

```
typedef struct _NSHeightInfo {
        float newHeight;
        float oldHeight;
        NSLineDesc   lineDesc;
} NSHeightInfo;
```

Stores height information for each line of text in an NSText object.

```
typedef struct _NSLay {
        float x;
        float y;
        short     offset;
```

Represents a single sequence of text in a line and records everything needed to select or draw that piece.

```
    short     chars;
    id    font;
    void     *paraStyle;
    NSRun *run;
    NSLayFlags lFlags;
} NSLay;
```

```
typedef struct _NSLayArray {
    NSTextChunk chunk;
    NSLay     lays[1];
} NSLayArray;
```

Holds the layout for the current line. Since the structure's first field is an **NSTextChunk** structure, **NSLayArray**s can be manipulated by the functions that manage variable-sized arrays of records.

```
typedef struct {
    unsigned int mustMove:1;
    unsigned int isMoveChar:1;
    unsigned int RESERVED:14;
} NSLayFlags;
```

Records whether a text lay in an NSText object needs special treatment (e.g., because of non-printing characters).

```
typedef struct _NSLayInfo {
    NSRect rect;
    float descent;
    float width;
    float left;
    float right;
    float rightIndent;
    NSLayArray *lays;
    NSWidthArray *widths;
    NSCharArray *chars;
    NSTextCache cache;
    NSRect *textClipRect;
    struct _lFlags {
        unsigned int horizCanGrow:1;
        unsigned int vertCanGrow:1;
        unsigned int erase:1;
        unsigned int ping:1;
        unsigned int endsParagraph:1;
        unsigned int resetCache:1;
```

NSText's scanning and drawing functions use this structure to communicate information about lines of text.

```
        unsigned int RESERVED:10;
    } lFlags;
} NSLayInfo;
```

```
typedef short NSLineDesc;
```

Used to identify lines of text in the NSText object.

```
typedef enum _NSParagraphProperty {
    NSLeftAlignedParagraph,
    NSRightAlignedParagraph,
    NSCenterAlignedParagraph,
    NSJustificationAlignedParagraph,
    NSFirstIndentParagraph,
    NSIndentParagraph,
    NSAddTabParagraph,
    NSRemoveTabParagraph,
    NSLeftMarginParagraph,
    NSRightMarginParagraph
} NSParagraphProperty;
```

The constants of this type identify specific paragraph properties for selected text. NSText's s**etSelProp:** method takes this argument type.

```
typedef struct _NSRun {
    id   font;
    int chars;
    void    *paraStyle;
    int   textRGBColor;
    unsigned char    superscript;
    unsigned char    subscript;
    id    info;
    NSRunFlags rFlags;
} NSRun;
```

In an NSText object, this structure represents a single sequence of text with a given format.

```
typedef struct _NSRunArray {
    NSTextChunk chunk;
    NSRun    runs[1];
} NSRunArray;
```

This structure holds the array of text runs in an NSText object. Since the first field is an NSTextChunk structure you can manipulate the items in the array with the functions that manage variable-sized arrays of records.

```
typedef struct {
    unsigned int underline:1;
    unsigned int dummy:1;
```

The fields of this structure record whether a run in an NSText object contains graphics, is underlined, or if an alternate character forced the use of a symbol.

```
        unsigned int subclassWantsRTF:1;
        unsigned int graphic:1;
        unsigned int forcedSymbol:1;
        unsigned int RESERVED:11;
} NSRunFlags;
```

```
typedef struct _NSSelPt {
        int cp;
        int line;
        float x;
        float y;
        int c1st;
        float ht;
} NSSelPt;
```

Represents one end of a selection in an NSText object.
- **cp** — Character position.
- **line** — Offset of LineDesc in break table.
- **x** — x coordinate.
- **y** — y coordinate.
- **c1st** — Character position of first character in the line.
- **ht** — Line height.

```
typedef struct _NSTabStop {
        short    kind;
        float x;
} NSTabStop;
```

This structure describes an NSText object's tab stops.

```
typedef struct _NSTextBlock {
        struct _NSTextBlock *next;
        struct _NSTextBlock *prior;
        struct _tbFlags {
                unsigned int malloced:1;
                unsigned int PAD:15;
        } tbFlags;
        short    chars;
        unsigned char *text;
} NSTextBlock;
```

A structure holds text characters in blocks no bigger than **NSTextBlockSize** (see below). A linked list of these text blocks comprises the text for an NSText object.

```
typedef struct _NSTextCache {
        int curPos;
        NSRun *curRun;
        int runFirstPos;
        NSTextBlock *curBlock;
        int blockFirstPos;
```

This structure describes the current text block and run, and the cursor position in the text.

```
} NSTextCache;
```

```
typedef struct _NSTextChunk {
        short     growby;
        int allocated;
        int used;
} NSTextChunk;
```

NSText uses this structure to implement variable-sized arrays of records.

```
typedef char    *(*NSTextFilterFunc) (
        id self,
        unsigned char * insertText,
        int *insertLength,
        int position);
```

A text filter function implements autoindenting and other features in an NSText object.

```
typedef int (*NSTextFunc) (
        id self,
        NSLayInfo *layInfo);
```

This is the type for an NSText object's scanning and drawing function, as set through the **setScanFunc:** and **setDrawFunc:** methods.

```
typedef enum _NSTextAlignment {
        NSLeftTextAlignment,
        NSRightTextAlignment,
        NSCenterTextAlignment,
        NSJustifiedTextAlignment,
        NSNaturalTextAlignment
} NSTextAlignment;
```

The constants of this type determine text alignment. Used by methods of NSCell, NSControl, NSForm, NSFormCell, and NSText. **NSNaturalTextAlignment** indicates the default alignment for the text.

```
typedef struct _NSTextStyle {
        float indent1st;
        float indent2nd;
        float lineHt;
        float descentLine;
        NSTextAlignment     alignment;
        short     numTabs;
        NSTabStop   *tabs;
} NSTextStyle;
```

NSText uses this structure to describe text layout and tab stops.

```
typedef struct _NSWidthArray {
        NSTextChunk chunk;
```

Holds the character widths for the current line. Since the first field is an NSTextChunk structure

```
    float widths[1];
} NSWidthArray;
```

you can manipulate the items in the array with the functions that manage variable-sized arrays of records.

```
enum {
    NSLeftTab
};
```

This constant is used by the NSText object's tab functions.

```
enum {
    NSBackspaceKey     = 8,
    NSCarriageReturnKey     = 13,
    NSDeleteKey= 0x7f,
    NSBacktabKey     = 25
};
```

These character-code constants are used by the NSText object's character filter function.

```
enum {
    NSIllegalTextMovement    = 0,
    NSReturnTextMovement    = 0x10,
    NSTabTextMovement     = 0x11,
    NSBacktabTextMovement    = 0x12,
    NSLeftTextMovement     = 0x13,
    NSRightTextMovement     = 0x14,
    NSUpTextMovement     = 0x15,
    NSDownTextMovement     = 0x16
};
```

Movement codes describing types of movement between text fields. Passed in toNSText delegates as the last argument of **textDidEnd:endChar:**.

```
enum {
    NSTextBlockSize     = 512
};
```

The size, in bytes, of a text block.

Break Tables

These tables (with their associated sizes) are finite-state machines that determine word wrapping in an NSText object.

const NSFSM ***NSCBreakTable**;

int **NSCBreakTableSize**;

const NSFSM ***NSEnglishBreakTable**;

int **NSEnglishBreakTableSize**;

const NSFSM ***NSEnglishNoBreakTable**;

int **NSEnglishNoBreakTableSize**;

Character Category Tables

These tables define the character classes used in an NSText object's break and click tables.

const unsigned char ***NSCCharCatTable**;

const unsigned char ***NSEnglishCharCatTable**;

Click Tables

NSText objects use these tables as finite-state machines that determine which characters are selected when the user double-clicks.

const NSFSM ***NSCClickTable**;

int **NSCClickTableSize**;

const NSFSM ***NSEnglishClickTable**;

int **NSEnglishClickTableSize**;

Smart Cut and Paste Tables

These tables are suitable as arguments for the NSText methods **setPreSelSmartable:** and **setPostSelSmartTable:**. When users paste text into an NSText object, if the character to the left (right) side of the new word is not in the left (right) table, an extra space is added to that side.

const unsigned char ***NSCSmartLeftChars**;

const unsigned char ***NSCSmartRightChars**;

const unsigned char ***NSEnglishSmartLeftChars**;

const unsigned char ***NSEnglishSmartRightChars**;

NSCStringText Internal State Structure

This is the structure returned by the **cStringTextInternalState** method of NSCStringText, for use only by applications that need to access the internal state of an NSCStringText object.

typedef struct _NSCStringTextInternalState {

const NSFSM ***breakTable**;	Pointer to state table that specifies word and line breaks
const NSFSM ***clickTable**;	Pointer to state table that defines word boundaries for double-click selection
const unsigned char ***preSelSmartTable**;	Pointer to table that specifies which characters on the left end of a selection are treated as equivalent to a space
const unsigned char ***postSelSmartTable**;	Pointer to table that specifies which characters on the right end of a selection are treated as equivalent to a space
const unsigned char ***charCategoryTable**;	Pointer to table that maps ASCII characters to character classes.
char **delegateMethods**;	Record of notification methods the delegate implements
NSCharFilterFunc **charFilterFunc**;	Function to check each character as it's typed into the text
NSTextFilterFunc **textFilterFunc**;	Function to check text that's being added to the NSCStringText object
NSString ***_string**;	Reserved for internal use
NSTextFunc **scanFunc**;	Function that calculates the line of text
NSTextFunc **drawFunc**;	Function that draws the line of text
id **delegate**;	Object that's notified when the NSCStringText object is modified
int **tag**;	Integer the delegate uses to identify the NSCStringText object
void ***cursorTE**;	Timed entry number for the vertical bar that marks the insertion point
NSTextBlock ***firstTextBlock**;	Pointer to first record in a linked list of text blocks
NSTextBlock ***lastTextBlock**;	Pointer to last record in a linked list of text blocks
NSRunArray ***theRuns**;	Pointer to array of format runs. By default, **theRuns** points to a single run of the default font
NSRun **typingRun**;	Format run to use for the next characters entered
NSBreakArray ***theBreaks**;	Pointer to the array of line breaks
int **growLine**;	Line containing the end of the growing selection
int **textLength**;	Number of characters in the NSCStringText object
float **maxY**;	Bottom of the last line of text, relative to the origin of **bodyRect**
float **maxX**;	Widest line of text. Only accurate after **calcLine** method is invoked
NSRect **bodyRect**;	Rectangle in which the NSCStringText object draws
float **borderWidth**;	Reserved for internal use
char **clickCount**;	Number of clicks that created the selection

NSSelPt **sp0**;	Starting position of the selection
NSSelPt **spN**;	Ending position of the selection
NSSelPt **anchorL**;	Left anchor position
NSSelPt **anchorR**;	Right anchor position
NSSize **maxSize**;	Maximum size of the frame rectangle
NSSize **minSize**;	Minimum size of the frame rectangle

```
struct _tFlags {
#ifdef __BIG_ENDIAN__
```

unsigned int **_editMode**:2;	Reserved for internal use
unsigned int **_selectMode**:2;	Reserved for internal use
unsigned int **_caretState**:2;	Reserved for internal use
unsigned int **changeState**:1;	True if any changes have been made to the text since the NSCStringText object became first responder
unsigned int **charWrap**:1;	True if the NSCStringText object wraps words whose length exceeds the line length on a character basis. False if such words are truncated at end of line
unsigned int **haveDown**:1;	True if the left mouse button (or any button if button functions are not differentiated) is down
unsigned int **anchorIs0**:1;	True if the anchor's position is at **sp0**
unsigned int **horizResizable**:1;	True if the NSCStringText object's width can grow or shrink
unsigned int **vertResizable**:1;	True if the NSCStringText object's height can grow or shrink
unsigned int **overstrikeDiacriticals**:1;	Reserved for internal use
unsigned int **monoFont**:1;	True if the NSCStringText object uses one font for all its text
unsigned int **disableFontPanel**:1;	True if the NSCStringText object doesn't update the font panel automatically
unsigned int **inClipView**:1;	True if the NSCStringText object is a subview of an NSClipView

```
#else
        unsigned int inClipView:1;
        unsigned int disableFontPanel:1;
        unsigned int monoFont:1;
        unsigned int overstrikeDiacriticals:1;
        unsigned int vertResizable:1;
        unsigned int horizResizable:1;
        unsigned int anchorIs0:1;
        unsigned int haveDown:1;
        unsigned int charWrap:1;
        unsigned int changeState:1;
        unsigned int _caretState:2;
```

```
        unsigned int _selectMode:2;
        unsigned int _editMode:2;
    #endif
    } tFlags;
    void *_info;                    Reserved for internal use
    void *_textStr;                 Reserved for internal use
} NSCStringTextInternalState;
```

View

```
typedef int NSTrackingRectTag;
```

A unique identifier of a tracking rectangle assigned by NSView. (See **addTrackingRectangle:owner: assumeInside:**.)

```
typedef enum _NSBorderType {
    NSNoBorder,
    NSLineBorder,
    NSBezelBorder,
    NSGrooveBorder
} NSBorderType;
```

Constants representing the four types of borders that can appear around NSView objects.

```
enum {
    NSViewNotSizable,
    NSViewMinXMargin,
    NSViewWidthSizable,
    NSViewMaxXMargin,
    NSViewMinYMargin,
    NSViewHeightSizable,
    NSViewMaxYMargin
};
```

NSView uses these autoresize constants to describe the parts of a view (or its margins) that are resized when the view's superview is resized.

Window

```
enum {
```

These constants list the window-device tiers that the

NSNormalWindowLevel = 0,
NSFloatingWindowLevel = 3,
NSDockWindowLevel = 5,
NSSubmenuWindowLevel = 10,
NSMainMenuWindowLevel = 20
};

Application Kit uses. Windows are ordered (or ªlayeredº) within tiers: The uppermost window in one tier can still be obscured by the lowest window in the next higher tier.

enum {
NSBorderlessWindowMask,
NSTitledWindowMask,
NSClosableWindowMask,
NSMiniaturizableWindowMask,
NSResizableWindowMask
};

Bitmap masks to determine certain window styles.

Size Globals

These global constants give the dimensions of an icon and contained.

NSSize **NSIconSize**;

NSSize **NSTokenSize**:

Workspace

Workspace File Type Globals

Identifies the type of file queried by the method **getInfoForFile:application:type:** (passed back by reference in last argument).

NSString ***NSPlainFileType**;

NSString ***NSDirectoryFileType**;

NSString ***NSApplicationFileType**;

NSString ***NSFilesystemFileType**;

NSString ***NSShellCommandFileType**;

Workspace File Operation Globals

Used as file-operation arguments in the **performFileOperation:source:destination:files:options:** method (first argument).

NSString ***NSWorkspaceCompressOperation**;

NSString ***NSWorkspaceCopyOperation**;

NSString ***NSWorkspaceDecompressOperation**;

NSString ***NSWorkspaceDecryptOperation**;

NSString ***NSWorkspaceDestroyOperation**;

NSString ***NSWorkspaceDuplicateOperation**;

NSString ***NSWorkspaceEncryptOperation**;

NSString ***NSWorkspaceLinkOperation**;

NSString ***NSWorkspaceMoveOperation**;

NSString ***NSWorkspaceRecycleOperation**;