NSWindow Class Reference

Cocoa > User Experience

2009-03-04

Apple Inc.
© 2009 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Carbon, Cocoa, ColorSync, Mac, Mac OS, and Quartz are trademarks of Apple Inc., registered in the United States and other countries.

Aperture is a trademark of Apple Inc.

NeXT and NeXTSTEP are trademarks of NeXT Software, Inc., registered in the United States and other countries.

Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Tables

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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.
Companion guide	Window Programming Guide for Cocoa
Declared in	NSDrawer.h NSGraphics.h NSWindow.h
Related sample code	CoreRecipes ImageClient MyPhoto Quartz Composer WWDC 2005 TextEdit TextEditPlus

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 68)
  Initializes the window with the specified values.
- `initWithContentRect:styleMask:backing:defer:screen:` (page 69)
  Initializes an allocated window with the specified values.

Configuring Windows

- `styleMask` (page 132)
  Returns the window's style mask, indicating what kinds of control items it displays.
- `worksWhenModal` (page 138)
  Indicates whether the window is able to receive keyboard and mouse events even when some other window is being run modally.
- `alphaValue` (page 36)
  Returns the window's alpha value.
- `setAlphaValue:` (page 100)
  Applies a given alpha value to the entire window.
- `backgroundColor` (page 39)
  Returns the color of the window's background.
- `setBackgroundColor:` (page 103)
  Sets the window's background color to the given color.
- `contentView` (page 49)
  Returns the window's content view, the highest accessible `NSView` object in the window's view hierarchy.
- `setContentView:` (page 108)
  Makes a given view the window's content view.
- `canHide` (page 42)
  Indicates whether the window can be hidden when its application becomes hidden (during execution of the `NSApplication hide:` method).
- `setCanHide:` (page 104)
  Specifies whether the window can be hidden when its application becomes hidden (during execution of the `NSApplication hide:` method).
- `hidesOnDeactivate` (page 67)
  Indicates whether the window is removed from the screen when its application becomes inactive.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- `setHidesOnDeactivate:` (page 117)
  Specifies whether the window is removed from the screen when the application is inactive.
- `collectionBehavior` (page 45)
  Identifies the window's behavior in window collections.
- `setCollectionBehavior:` (page 105)
  Specifies the window's behavior in window collections.
- `isOpaque` (page 75)
  Indicates whether the window is opaque.
- `setOpaque:` (page 122)
  Specifies whether the window is opaque.
- `hasShadow` (page 66)
  Indicates whether the window has a shadow.
- `setHasShadow:` (page 117)
  Specifies whether the window has a shadow.
- `invalidateShadow` (page 71)
  Invalidates the window shadow so that it is recomputed based on the current window shape.
- `autorecalculatesContentBorderThicknessForEdge:` (page 38)
  Indicates whether the window calculates the thickness of a given border automatically.
- `setAutorecalculatesContentBorderThickness:forEdge:` (page 102)
  Specifies whether the window calculates the thickness of a given border automatically.
- `contentBorderThicknessForEdge:` (page 47)
  Indicates the thickness of a given border of the window.
- `setContentBorderThickness:forEdge:` (page 106)
  Specifies the thickness of a given border of the window.
- `delegate` (page 52)
  Returns the window's delegate.
- `setDelegate:` (page 109)
  Sets the window's delegate to a given object or removes an existing delegate.
- `canBeVisibleOnAllSpaces` (page 167) 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.
- `setCanBeVisibleOnAllSpaces:` (page 167) 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.

Accessing Window Information

+ `defaultDepthLimit` (page 31)
  Returns the default depth limit for instances of `NSWindow`.
- `windowNumber` (page 137)
  Provides the window number of the window's window device.
- `gState` (page 66)
  Returns the window's graphics state object.
- `canStoreColor` (page 43)
  Indicates whether the window has a depth limit that allows it to store color values.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- `deviceDescription` (page 54)
  Returns a dictionary containing information about the window's resolution.
- `canBecomeVisibleWithoutLogin` (page 42)
  Indicates whether the window can be displayed at the log-in window. Default: `NO`.
- `setCanBecomeVisibleWithoutLogin:` (page 104)
  Specifies whether the window can be displayed at the login window.
- `sharingType` (page 130)
  Indicates the level of access other processes have to the window's content.
- `setSharingType:` (page 126)
  Specifies the level of access other processes have to the window's content.
- `backingType` (page 39)
  Returns the window's backing store type.
- `setBackingType:` (page 103)
  Sets the window's backing store type to a given type.
- `backingLocation` (page 39)
  Indicates the window's backing store location.
- `preferredBackingLocation` (page 90)
  Indicates the preferred location for the window's backing store.
- `setPreferredBackingLocation:` (page 123)
  Specifies the preferred location for the window's backing store.
- `isOneShot` (page 74)
  Indicates whether the window device the window manages is freed when it's removed from the screen list.
- `setOneShot:` (page 122)
  Sets whether the window device that the window manages should be freed when it's removed from the screen list.
- `depthLimit` (page 53)
  Returns the depth limit of the window.
- `setDepthLimit:` (page 110)
  Sets the depth limit of the window to a given limit.
- `hasDynamicDepthLimit` (page 66)
  Indicates whether the window's depth limit can change to match the depth of the screen it's on.
- `setDynamicDepthLimit:` (page 111)
  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 31)
  Returns the content rectangle used by a window with a given frame rectangle and window style.
+ `frameRectForContentRect:styleMask:` (page 32)
  Returns the frame rectangle used by a window with a given content rectangle and window style.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

+ `minFrameWidthWithTitle:styleMask:` (page 33)
  Returns the minimum width a window's frame rectangle must have for it to display a title, with a given window style.

- `contentRectForFrameRect:` (page 48)
  Returns the window's content rectangle with a given frame rectangle.

- `frameRectForContentRect:` (page 65)
  Returns the window's frame rectangle with a given content rectangle.

Managing Windows

- `drawers` (page 59)
  Returns the collection of drawers associated with the window.

- `windowController` (page 136)
  Returns the window's window controller.

- `setWindowController:` (page 129)
  Sets the window's window controller.

Managing Sheets

- `attachedSheet` (page 37)
  Returns the sheet attached to the window.

- `isSheet` (page 76)
  Indicates whether the window has ever run as a modal sheet.

- `window:willPositionSheet:usingRect:` (page 141) *delegate method*
  Sent to the delegate just before the animation of a sheet, giving it the opportunity to return a custom location for the attachment of a sheet to a window.

- `windowWillBeginSheet:` (page 147) *delegate method*
  Sent by the default notification center immediately before an `NSWindow` object opens a sheet.

- `windowDidEndSheet:` (page 143) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object closes a sheet.

Sizing

- `frame` (page 64)
  Returns the window's frame rectangle.

- `setFrameOrigin:` (page 115)
  Positions the bottom-left corner of the window's frame rectangle at a given point in screen coordinates.

- `setFrameTopLeftPoint:` (page 115)
  Positions the top-left corner of the window's frame rectangle at a given point in screen coordinates.

- `constrainFrameRect:toScreen:` (page 46)
  Modifies and returns a frame rectangle so that its top edge lies on a specific screen.

- `cascadeTopLeftFromPoint:` (page 43)
  Positions the window's top left to a given point.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- `setFrame:display:` (page 112)
  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 113)
  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 36)
  Specifies the duration of a smooth frame-size change.
- `windowWillUseStandardFrame:defaultFrame:` (page 150) *delegate method*
  Invoked by the `zoom:` (page 138) method while determining a frame an `NSWindow` object may be zoomed to.
- `aspectRatio` (page 37)
  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 101)
  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 82)
  Returns the minimum size to which the window's frame (including its title bar) can be sized.
- `setMinSize:` (page 121)
  Sets the minimum size to which the window's frame (including its title bar) can be sized to *`aSize`*.
- `maxSize` (page 80)
  Returns the maximum size to which the window's frame (including its title bar) can be sized.
- `setMaxSize:` (page 119)
  Sets the maximum size to which the window's frame (including its title bar) can be sized.
- `isZoomed` (page 76)
  Returns a Boolean value that indicates whether the window is in a zoomed state.
- `performZoom:` (page 89)
  This action method simulates the user clicking the zoom box by momentarily highlighting the button and then zooming the window.
- `zoom:` (page 138)
  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).
- `windowShouldZoom:toFrame:` (page 147) *delegate method*
  Sent just before *`sender`* is zoomed to allow or disallow the operation.
- `resizeFlags` (page 95)
  Returns the flags field of the event record for the mouse-down event that initiated the resizing session.
- `showsResizeIndicator` (page 130)
  Returns a Boolean value that indicates whether the window's resize indicator is visible.
- `setShowsResizeIndicator:` (page 126)
  Specifies whether the window's resize indicator is visible
- `resizeIncrements` (page 95)
  Returns the window's resizing increments.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- `setResizeIncrements:` (page 126)
  Restricts the user's ability to resize the window so the width and height change by multiples of width and height increments.
- `preservesContentDuringLiveResize` (page 90)
  Returns a Boolean value that indicates whether the window tries to optimize live resize operations by preserving the content of views that have not changed.
- `setPreservesContentDuringLiveResize:` (page 124)
  Specifies whether the window tries to optimize live resize operations by preserving the content of views that have not changed.
- `windowWillResize:toSize:` (page 148) *delegate method*
  Invoked when a window is being resized (whether by the user or through one of the `setFrame...` methods other than `setFrame:display:` (page 112)).
- `windowDidResize:` (page 145) *delegate method*
  Sent by the default notification center immediately after a window has been resized.

Sizing Content

- `contentAspectRatio` (page 46)
  Returns the window's content aspect ratio.
- `setContentAspectRatio:` (page 105)
  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 48)
  Returns the minimum size of the window's content view.
- `setContentMinSize:` (page 106)
  Sets the minimum size of the window's content view in the window's base coordinate system.
- `setContentSize:` (page 107)
  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 47)
  Returns the maximum size of the window's content view.
- `setContentMaxSize:` (page 106)
  Sets the maximum size of the window's content view in the window's base coordinate system.
- `contentResizeIncrements` (page 48)
  Returns the window's content-view resizing increments.
- `setContentResizeIncrements:` (page 107)
  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 76)
  Indicates whether the window is visible onscreen (even when It's obscured by other windows).

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- `orderOut:` (page 86)
  Removes the window from the screen list, which hides the window.
- `orderBack:` (page 84)
  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 85)
  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 85)
  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 86)
  Repositions the window's window device in the window server's screen list.
- `level` (page 78)
  Returns the window level of the window.
- `setLevel:` (page 118)
  Sets the window's window level to a given level.

Managing Window Frames in User Defaults

+ `removeFrameUsingName:` (page 33)
  Removes the frame data stored under a given name from the application's user defaults.
- `setFrameUsingName:` (page 116)
  Sets the window's frame rectangle by reading the rectangle data stored under a given name from the defaults system.
- `setFrameUsingName:force:` (page 116)
  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 96)
  Saves the window's frame rectangle in the user defaults system under a given name.
- `frameAutosaveName` (page 64)
  Returns the name used to automatically save the window's frame rectangle data in the defaults system, as set through `setFrameAutosaveName:` (page 113).
- `setFrameAutosaveName:` (page 113)
  Sets the name used to automatically save the window's frame rectangle in the defaults system to a given name.
- `stringWithSavedFrame` (page 131)
  Returns a string representation of the window's frame rectangle.
- `setFrameFromString:` (page 114)
  Sets the window's frame rectangle from a given string representation.

Managing Key Status

- `isKeyWindow` (page 73)
  Indicates whether the window is the key window for the application.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- `canBecomeKeyWindow` (page 41)
  Indicates whether the window can become the key window.
- `makeKeyWindow` (page 80)
  Makes the window the key window.
- `makeKeyAndOrderFront:` (page 79)
  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 40)
  Invoked automatically to inform the window that it has become the key window; never invoke this method directly.
- `windowDidBecomeKey:` (page 141) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object has become key.
- `resignKeyWindow` (page 94)
  Invoked automatically when the window resigns key window status; never invoke this method directly.
- `windowDidResignKey:` (page 145) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object has resigned its status as key window.

Managing Main Status

- `isMainWindow` (page 73)
  Indicates whether the window is the application's main window.
- `canBecomeMainWindow` (page 42)
  Indicates whether the window can become the application's main window.
- `makeMainWindow` (page 80)
  Makes the window the main window.
- `becomeMainWindow` (page 40)
  Invoked automatically to inform the window that it has become the main window; never invoke this method directly.
- `windowDidBecomeMain:` (page 142) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object has become main.
- `resignMainWindow` (page 94)
  Invoked automatically when the window resigns main window status; never invoke this method directly.
- `windowDidResignMain:` (page 145) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object has resigned its status as main window.

Managing Toolbars

- `toolbar` (page 133)
  Returns the window's toolbar.
- `setToolbar:` (page 128)
  Sets the window's toolbar.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- `toggleToolbarShown:` (page 132)
  The action method for the “Hide Toolbar” menu item (which alternates with “Show Toolbar”).
- `runToolbarCustomizationPalette:` (page 96)
  The action method for the “Customize Toolbar…” menu item.

Managing Attached Windows

- `childWindows` (page 44)
  Returns an array of the window’s attached child windows.
- `addChildWindow:ordered:` (page 34)
  Adds a given window as a child window of the window.
- `removeChildWindow:` (page 92)
  Detaches a given child window from the window.
- `parentWindow` (page 87)
  Returns the parent window to which the window is attached as a child.
- `setParentWindow:` (page 123)
  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 73)
  Indicates whether the window’s flushing ability is disabled.
- `enableFlushWindow` (page 60)
  Reenables the `flushWindow` (page 63) method for the window after it was disabled through a previous `disableFlushWindow` (page 55) message.
- `disableFlushWindow` (page 55)
  Disables the `flushWindow` (page 63) method for the window.
- `flushWindow` (page 63)
  Flushes the window’s offscreen buffer to the screen if the window is buffered and flushing is enabled.
- `flushWindowIfNeeded` (page 64)
  Flushes the window’s offscreen buffer to the screen if flushing is enabled and if the last `flushWindow` (page 63) message had no effect because flushing was disabled.

Managing Default Buttons

- `defaultButtonCell` (page 52)
  Returns the button cell that performs as if clicked when the window receives a Return (or Enter) key event.
- `setDefaultButtonCell:` (page 109)
  Makes the key equivalent of button cell the Return (or Enter) key, so when the user presses Return that button performs as if clicked.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- `enableKeyEquivalentForDefaultButtonCell` (page 60)
  Reenables the default button cell's key equivalent, so it performs a click when the user presses Return (or Enter).
- `disableKeyEquivalentForDefaultButtonCell` (page 55)
  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 61)
  Returns the window's field editor, creating it if requested.
- `windowWillReturnFieldEditor:toObject:` (page 149) *delegate method*
  Invoked when the field editor for a text-displaying object is requested.
- `endEditingFor:` (page 61)
  Forces the field editor to give up its first responder status and prepares it for its next assignment.

Managing the Window Menu

- `isExcludedFromWindowsMenu` (page 72)
  Indicates whether the window is excluded from the application's Windows menu.
- `setExcludedFromWindowsMenu:` (page 112)
  Specifies whether the window's title is omitted from the application's Windows menu.

Managing Cursor Rectangles

- `areCursorRectsEnabled` (page 37)
  Indicates whether the window's cursor rectangles are enabled.
- `enableCursorRects` (page 60)
  Reenables cursor rectangle management within the window after a `disableCursorRects` (page 54) message.
- `disableCursorRects` (page 54)
  Disables all cursor rectangle management within the window.
- `discardCursorRects` (page 56)
  Invalidates all cursor rectangles in the window.
- `invalidateCursorRectsForView:` (page 71)
  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 93)
  Clears the window's cursor rectangles and the cursor rectangles of the `NSView` objects in its view hierarchy.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Managing Title Bars

+ `standardWindowButton:forStyleMask:` (page 33)
  Returns a new instance of a given standard window button, sized appropriately for a given window style.

- `standardWindowButton:` (page 131)
  Returns the window button of a given window button kind in the window's view hierarchy.

- `showsToolbarButton` (page 130)
  Indicates whether the toolbar control button is currently displayed.

- `setShowsToolbarButton:` (page 127)
  Specifies whether the window shows the toolbar control button.

Managing Tooltips

- `allowsToolTipsWhenApplicationIsInactive` (page 35)
  Indicates whether the window can display tooltips even when the application is in the background.

- `setAllowsToolTipsWhenApplicationIsInactive:` (page 100)
  Specifies whether the window can display tooltips even when the application is in the background.

Handling Events

+ `menuChanged:` (page 32)
  This method does nothing; it is here for backward compatibility.

- `currentEvent` (page 50)
  Returns the event currently being processed by the application, by invoking `NSApplication`'s `currentEvent` method.

- `nextEventMatchingMask:` (page 83)
  Returns the next event matching a given mask.

- `nextEventMatchingMask:untilDate:inMode:dequeue:` (page 83)
  Forwards the message to the global `NSApplication` object, `NSApp`.

- `discardEventsMatchingMask:beforeEvent:` (page 56)
  Forwards the message to the `NSApplication` object, `NSApp`.

- `postEvent:atStart:` (page 89)
  Forwards the message to the global `NSApplication` object, `NSApp`.

- `sendEvent:` (page 99)
  This action method dispatches mouse and keyboard events sent to the window by the `NSApplication` object.

- `tryToPerform:with:` (page 133)
  Dispatches action messages with a given argument.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Managing Responders

- `initialFirstResponder` (page 68)
  Returns view that's made first responder the first time the window is placed onscreen.
- `firstResponder` (page 62)
  Returns the window's first responder.
- `setInitialFirstResponder:` (page 118)
  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 78)
  Attempts to make a given responder the first responder for the window.

Managing the Key View Loop

- `selectKeyViewPrecedingView:` (page 98)
  Makes key the view that precedes the given view.
- `selectKeyViewFollowingView:` (page 97)
  Makes key the view that follows the given view.
- `selectPreviousKeyView:` (page 99)
  This action method searches for a candidate previous key view and, if it finds one, invokes `makeFirstResponder:` (page 78) to establish it as the first responder.
- `selectNextKeyView:` (page 98)
  This action method searches for a candidate next key view and, if it finds one, invokes `makeFirstResponder:` (page 78) to establish it as the first responder.
- `keyViewSelectionDirection` (page 77)
  Returns the direction the window is currently using to change the key view.
- `autorecalculatesKeyViewLoop` (page 38)
  Indicates whether the window automatically recalculates the key view loop when views are added.
- `recalculateKeyViewLoop` (page 91)
  Marks the key view loop as dirty and in need of recalculation.
- `setAutorecalculatesKeyViewLoop:` (page 103)
  Specifies whether to recalculate the key view loop automatically when views are added or removed.

Handling Keyboard Events

- `keyDown:` (page 77)
  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 34)
  Indicates whether the window accepts mouse-moved events.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- `ignoresMouseEvents` (page 67)
  Indicates whether the window is transparent to mouse events.
- `setIgnoresMouseEvents:` (page 118)
  Specifies whether the window is transparent to mouse clicks and other mouse events, allowing overlay windows.
- `mouseLocationOutsideOfEventStream` (page 82)
  Returns the current location of the pointer reckoned in the window's base coordinate system.
- `setAcceptsMouseMovedEvents:` (page 100)
  Specifies whether the window is to accept mouse-moved events.

Bracketing Drawing Operations

- `cacheImageInRect:` (page 40)
  Stores the window's raster image from a given rectangle expressed in the window's base coordinate system.
- `restoreCachedImage` (page 95)
  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 40).
- `discardCachedImage` (page 56)
  Discards all of the window's cached image rectangles.

Drawing Windows

- `display` (page 57)
  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 57)
  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 136)
  Indicates whether any of the window's views need to be displayed.
- `setViewsNeedDisplay:` (page 129)
  Specifies whether the window's views need to be displayed..
- `isAutodisplay` (page 71)
  Indicates whether the window automatically displays views that need to be displayed.
- `setAutodisplay:` (page 101)
  Specifies whether the window is to automatically display the views that are marked as needing it.
- `useOptimizedDrawing:` (page 135)
  Specifies whether the window is to optimize focusing and drawing when displaying its views.
- `graphicsContext` (page 65)
  Provides the graphics context associated with the window for the current thread.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Updating Windows

- `disableScreenUpdatesUntilFlush` (page 55)
  Disables the window's screen updates until the window is flushed.
- `update` (page 134)
  Updates the window.
- `windowDidUpdate:` (page 146) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object receives an `update` (page 134) message.

Exposing Windows

- `windowDidExpose:` (page 144) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object has been exposed.

Dragging

- `dragImage:at:offset:event:pasteboard:source:slideBack:` (page 59)
  Begins a dragging session.
- `registerForDraggedTypes:` (page 91)
  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 134)
  Unregisters the window as a possible destination for dragging operations.
- `window:shouldDragDocumentWithEvent:from:withPasteboard:` (page 139) *delegate method*
  Determines whether the sender allows the user to drag the sender's represented file's icon from the sender's title bar.

Converting Coordinates

- `convertBaseToScreen:` (page 49)
  Converts a given point from the window's base coordinate system to the screen coordinate system.
- `convertScreenToBase:` (page 50)
  Converts a given point from the screen coordinate system to the window's base coordinate system.
- `userSpaceScaleFactor` (page 135)
  Returns the scale factor applied to the window.

Getting the Undo Manager

- `windowWillReturnUndoManager:` (page 150) *delegate method*
  Invoked when the undo manager for a window is requested. Returns the appropriate undo manager for the window.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Accessing Edited Status

- `isDocumentEdited` (page 72)
  Indicates whether the window's document has been edited.
- `setDocumentEdited:` (page 111)
  Specifies whether the window's document has been edited.

Managing Titles

- `title` (page 132)
  Returns either the string that appears in the title bar of the window, or the path to the represented file.
- `setTitle:` (page 127)
  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 128)
  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 125).
- `representedFilename` (page 92)
  Returns the pathname of the file the window represents.
- `setRepresentedFilename:` (page 125)
  Sets the pathname of the file the window represents.
- `representedURL` (page 93)
  Provides the URL of the file the window represents.
- `setRepresentedURL:` (page 125)
  Specifies the URL of the file the window represents.
- `window:shouldPopUpDocumentPathMenu:` (page 140) *delegate method*
  Determines whether the sender displays the title pop-up menu in response to a Command-click on the sender's title.

Accessing Screen Information

- `screen` (page 97)
  Returns the screen the window is on.
- `deepestScreen` (page 51)
  Returns the deepest screen the window is on (it may be split over several screens).
- `displaysWhenScreenProfileChanges` (page 58)
  Indicates whether the window context should be updated when the screen profile changes or when the window moves to a different screen.
- `setDisplaysWhenScreenProfileChanges:` (page 110)
  Specifies whether the window context should be updated when the screen profile changes.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Moving

- `isMovableByWindowBackground` (page 74)
  Indicates whether the window is movable by clicking and dragging anywhere in its background.
- `setMovableByWindowBackground:` (page 121)
  Sets whether the window is movable by clicking and dragging anywhere in its background.
- `center` (page 44)
  Sets the window's location to the center of the screen.
- `windowWillMove:` (page 148) *delegate method*
  Sent by the default notification center immediately before an `NSWindow` object is moved.
- `windowDidMove:` (page 144) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object has been moved.
- `windowDidChangeScreen:` (page 142) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object has changed screens.
- `windowDidChangeScreenProfile:` (page 142) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object has changed screen display profiles.

Closing Windows

- `performClose:` (page 88)
  This action method simulates the user clicking the close button by momentarily highlighting the button and then closing the window.
- `close` (page 45)
  Removes the window from the screen.
- `windowShouldClose:` (page 146) *delegate method*
  Invoked when the user attempts to close a window or a window receives a `performClose:` (page 88) message.
- `windowWillClose:` (page 147) *delegate method*
  Sent by the default notification center immediately before an `NSWindow` object closes.
- `isReleasedWhenClosed` (page 75)
  Indicates whether the window is released when it receives the `close` message.
- `setReleasedWhenClosed:` (page 124)
  Specifies whether the window is released when it receives the `close` message.

Minimizing Windows

- `isMiniaturized` (page 74)
  Indicates whether the window is minimized.
- `performMiniaturize:` (page 88)
  Simulates the user clicking the minimize button by momentarily highlighting the button, then minimizing the window.
- `miniaturize:` (page 81)
  Removes the window from the screen list and displays the minimized window in the Dock.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- `windowWillMiniaturize:` (page 148) *delegate method*
  Sent by the default notification center immediately before an `NSWindow` object is minimized.
- `windowDidMiniaturize:` (page 144) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object has been minimized.
- `deminiaturize:` (page 53)
  Deminimizes the window.
- `windowDidDeminiaturize:` (page 143) *delegate method*
  Sent by the default notification center immediately after an `NSWindow` object has been deminimized.
- `miniwindowImage` (page 81)
  Returns the custom miniaturized window image of the window.
- `setMiniwindowImage:` (page 120)
  Sets the window's custom minimized window image to a given image.
- `miniwindowTitle` (page 82)
  Returns the title displayed in the window's minimized window.
- `setMiniwindowTitle:` (page 120)
  Sets the title of the window's miniaturized counterpart to a given string and redisplays it.

Getting the Dock Tile

- `dockTile` (page 58)
  Provides the application's Dock tile.

Printing

- `print:` (page 90)
  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 50)
  Returns EPS data that draws the region of the window within a given rectangle.
- `dataWithPDFInsideRect:` (page 51)
  Returns PDF data that draws the region of the window within a given rectangle.

Providing Services

- `validRequestorForSendType:returnType:` (page 135)
  Searches for an object that responds to a Services request.

Working with Carbon

- `initWithWindowRef:` (page 70)
  Returns a Cocoa window created from a Carbon window.
- `windowRef` (page 137)
  Returns the Carbon `WindowRef` associated with the window, creating one if necessary.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 151) 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 48) instead of this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ frameRectForContentRect:styleMask:` (page 32)

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`, `NSColorSpaceFromDepth`, `NSBitsPerSampleFromDepth`, and `NSBitsPerPixelFromDepth`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setDepthLimit:` (page 110)

`- setDynamicDepthLimit:` (page 111)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- canStoreColor (page 43)

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 "Constants" (page 151) 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 65) instead of this method.

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

See Also
+ contentRectForFrameRect:styleMask: (page 31)

Related Sample Code
CocoaDragAndDrop

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 (NSResponder)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 "Constants" (page 151) 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 116)
- setFrameAutosaveName: (page 113)

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
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 151) 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 131)

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; `NO` otherwise.

Discussion

The `NSWindow` default is `NO`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setAcceptsMouseMovedEvents:` (page 100)

Declared In

`NSWindow.h`

addChildWindow:ordered:

Adds a given window as a child window of the window.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
- (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 92)
- `childWindows` (page 44)
- `parentWindow` (page 87)
- `setParentWindow:` (page 123)

Related Sample Code

GLChildWindowDemo

TrackBall

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; `NO` otherwise.

Discussion

The default is `NO`.

Availability

Available in Mac OS X v10.3 and later.

See Also

- `setAllowsToolTipsWhenApplicationIsInactive:` (page 100)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 100)

Related Sample Code
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 113).

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 differ in different releases of Mac OS X).

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

Declared In
NSWindow.h

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

areCursorRectsEnabled

Indicates whether the window's cursor rectangles are enabled.

```
- (BOOL)areCursorRectsEnabled
```

Return Value
`YES` when cursor rectangles are enabled, `NO` otherwise.

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

See Also
- `enableCursorRects` (page 60)
- `addCursorRect:cursor:` (`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 95)
- `setAspectRatio:` (page 101)
- `setFrame:display:` (page 112)

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 autorecalculates the given border's thickness; `NO` otherwise.

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

See Also
- `setAutorecalculatesContentBorderThickness:forEdge:` (page 102)

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; `NO` otherwise.

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

See Also
- `recalculateKeyViewLoop` (page 91)
- `setAutorecalculatesKeyViewLoop:` (page 103)

Declared In
`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 103)

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 "Constants" (page 151) for possible values.

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

See Also
- preferredBackingLocation (page 90)

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 "Constants" (page 151).

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

See Also
- setBackingType: (page 103)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 161) to the default notification center.

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

See Also
- `makeKeyWindow` (page 80)
- `makeKeyAndOrderFront:` (page 79)
- `becomeMainWindow` (page 40)

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 162) to the default notification center.

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

See Also
- `makeMainWindow` (page 80)
- `becomeKeyWindow` (page 40)

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
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 95) and `flushWindowIfNeeded` (page 64). The next time the window displays, it discards its cached image rectangles. You can also explicitly use `discardCachedImage` (page 56) 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 57)

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, `NO` otherwise.

Discussion

Attempts to make the window the key window are abandoned if this method returns `NO`. The `NSWindow` (page 13) implementation returns `YES` if the window has a title bar or a resize bar, `NO` otherwise.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- isKeyWindow` (page 73)

`- makeKeyWindow` (page 80)

Related Sample Code

CIAnnotation

FancyAbout

FunkyOverlayWindow

RoundTransparentWindow

TrackBall

Declared In

`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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; `NO` otherwise.

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 73)
- `makeMainWindow` (page 80)

Related Sample Code
FancyAbout

Declared In
`NSWindow.h`

canBecomeVisibleWithoutLogin

Indicates whether the window can be displayed at the log-in window. Default: `NO`.

```
- (BOOL)canBecomeVisibleWithoutLogin
```

Return Value
`YES` when the window can be displayed at the log-in window; `NO` otherwise.

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

See Also
- `setCanBecomeVisibleWithoutLogin:` (page 104)

Declared In
`NSWindow.h`

canHide

Indicates whether the window can be hidden when its application becomes hidden (during execution of the `NSApplication hide:` method).

```
- (BOOL)canHide
```

Return Value
`YES` if the window can be hidden when its application becomes hidden; `NO` otherwise.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Discussion
The default is `YES`.

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

See Also
`- setCanHide:` (page 104)

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; `NO` otherwise.

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

See Also
`- depthLimit` (page 53)
`- shouldDrawColor` (`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 115)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 79) 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:` method of the `NSApplication` class.

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

Related Sample Code
CocoaVideoFrameToGWorld
VertexPerformanceTest

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 92)
- `addChildWindow:ordered:` (page 34)
- `parentWindow` (page 87)
- `setParentWindow:` (page 123)

Declared In
NSWindow.h

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 124) 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 165) notification to the default notification center.

The close method differs in two important ways from the `performClose:` (page 88) method:

- It does not attempt to send a `windowShouldClose:` (page 146) 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 88) if you need these features.

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

See Also
`- orderOut:` (page 86)

Related Sample Code
ExtractMovieAudioToAIFF
QTExtractAndConvertToAIFF
QTExtractAndConvertToMovieFile
SillyFrequencyLevels
UIElementInspector

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

See Also
- `setCollectionBehavior:` (page 105)

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)`.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

See Also
- `setContentAspectRatio:` (page 105)

Declared In
`NSWindow.h`

contentBorderThicknessForEdge:

Indicates the thickness of a given border of the window.

```
- (CGFloat)contentBorderThicknessForEdge:(NSRectEdge)edge
```

Parameters

edge

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 106)

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 106)
- `contentMinSize` (page 48)

Declared In
`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 106)
- `contentMaxSize` (page 47)

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 "Constants" (page 151) for a list of style mask values. The main advantage of this instance-method counterpart to `contentRectForFrameRect:styleMask:` (page 31) 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 65)
+ `contentRectForFrameRect:styleMask:` (page 31)

Declared In
`NSWindow.h`

contentResizeIncrements

Returns the window's content-view resizing increments.

```
- (NSSize)contentResizeIncrements
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Return Value
The window's content-view resizing increments.

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

See Also
- `setContentResizeIncrements:` (page 107)

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 108)

Related Sample Code
CocoaDVDPlayer
FunkyOverlayWindow
LiveVideoMixer2
VideoViewer
WhackedTV

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

See Also
- `convertScreenToBase:` (page 50)
- `convertPoint:toView:` (`NSView`)

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 49)
- `convertRect:fromView:` (`NSView`)

Declared In
`NSWindow.h`

currentEvent

Returns the event currently being processed by the application, by invoking `NSApplication`'s `currentEvent` 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
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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:` (`NSView`)
- `writeEPSInsideRect:toPasteboard:` (`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:` (`NSView`)
- `writePDFInsideRect:toPasteboard:` (`NSView`)

Declared In
`NSWindow.h`

deepestScreen

Returns the deepest screen the window is on (it may be split over several screens).

```
- (NSScreen *)deepestScreen
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 97)
+ `deepestScreen` (`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 109)
- `disableKeyEquivalentForDefaultButtonCell` (page 55)
- `enableKeyEquivalentForDefaultButtonCell` (page 60)

Declared In
`NSWindow.h`

delegate

Returns the window's delegate.

```
- (id)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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

See Also
- `setDelegate:` (page 109)

Related Sample Code
FancyAbout
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

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 81)
- `styleMask` (page 132)

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`, `NSColorSpaceFromDepth`, `NSBitsPerSampleFromDepth`, and `NSBitsPerPixelFromDepth`.

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

See Also
- `+ defaultDepthLimit` (page 31)
- `- setDepthLimit:` (page 110)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- setDynamicDepthLimit: (page 111)

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 153).

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

See Also
- deviceDescription (NSScreen)
- bestRepresentationForDevice: (NSImage)
- colorUsingColorSpaceName: (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 60)

Declared In
NSWindow.h

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

disableFlushWindow

Disables the `flushWindow` (page 63) method for the window.

```
- (void)disableFlushWindow
```

Discussion
If the window is buffered, disabling `flushWindow` (page 63) 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 60) message. Invocations of these methods can be nested; flushing isn't reenabled until the last (unnested) `enableFlushWindow` (page 60) 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 52)
- `enableKeyEquivalentForDefaultButtonCell` (page 60)

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` function and reenables screen updates when the window flushes. Sending this message multiple times during a window update cycle has no effect.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

Related Sample Code
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 40)
- restoreCachedImage (page 95)
- display (page 57)

Declared In
NSWindow.h

discardCursorRects

Invalidates all cursor rectangles in the window.

```
- (void)discardCursorRects
```

Discussion
This method is invoked by resetCursorRects (page 93) 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
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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: (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 (NSView)
- displayIfNeeded (page 57)
- isAutodisplay (page 71)

Related Sample Code
GLChildWindowDemo

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 57)
- `displayIfNeeded` (`NSView`)
- `setNeedsDisplay:` (`NSView`)
- `isAutodisplay` (page 71)

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; `NO` otherwise.

Discussion
The default value is `NO`.

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

See Also
- `setDisplaysWhenScreenProfileChanges:` (page 110)

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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, NO to specify otherwise.

Discussion
This method should be invoked only from within a view's implementation of the mouseDown: or mouseDragged: 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: (NSView)

Declared In
NSWindow.h

drawers

Returns the collection of drawers associated with the window.

```
- (NSArray *)drawers
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 54) message.

```
- (void)enableCursorRects
```

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

Declared In
NSWindow.h

enableFlushWindow

Reenables the `flushWindow` (page 63) method for the window after it was disabled through a previous `disableFlushWindow` (page 55) 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 52)
- `disableKeyEquivalentForDefaultButtonCell` (page 55)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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` 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 78), 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 61) 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 61)
- `windowWillReturnFieldEditor:toObject:` (page 149)

Declared In
NSWindow.h

fieldEditor:forObject:

Returns the window's field editor, creating it if requested.

```
- (NSText *)fieldEditor:(BOOL)createWhenNeeded forObject:(id)anObject
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 149)) 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 61)). 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 61).

The window's delegate can substitute a custom field editor in place of the window's field editor by implementing windowWillReturnFieldEditor:toObject: (page 149). 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 149) 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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
- (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."

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

See Also
- `makeFirstResponder:` (page 78)
- `acceptsFirstResponder` (`NSResponder`)

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 57) and `displayIfNeeded` (page 57) methods and the corresponding `NSView` `display` and `displayIfNeeded` methods.

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

See Also
- `flushWindowIfNeeded` (page 64)
- `disableFlushWindow` (page 55)
- `enableFlushWindow` (page 60)

Related Sample Code
GLChildWindowDemo

Declared In
`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

flushWindowIfNeeded

Flushes the window's offscreen buffer to the screen if flushing is enabled and if the last `flushWindow` (page 63) message had no effect because flushing was disabled.

```
- (void)flushWindowIfNeeded
```

Discussion
To avoid unnecessary flushing, use this method rather than `flushWindow` (page 63) to flush an `NSWindow` object after flushing has been reenabled.

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

See Also
- `disableFlushWindow` (page 55)
- `enableFlushWindow` (page 60)

Declared In
`NSWindow.h`

frame

Returns the window's frame rectangle.

```
- (NSRect)frame
```

Return Value
The frame rectangle of the window in screen coordinates.

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

See Also
- `screen` (page 97)
- `deepestScreen` (page 51)

Related Sample Code
BasicCocoaAnimations
FunkyOverlayWindow
iSpend
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

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 113).

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
- (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 116)

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 "Constants" (page 151) for a list of style mask values. The major advantage of this instance-method counterpart to `frameRectForContentRect:styleMask:` (page 32) 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 48)
+ `frameRectForContentRect:styleMask:` (page 32)

Related Sample Code
BasicCocoaAnimations

Declared In
`NSWindow.h`

graphicsContext

Provides the graphics context associated with the window for the current thread.

```
- (NSGraphicsContext *)graphicsContext
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Return Value
The graphics context associated with the window for the current thread.

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

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.

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; NO otherwise.

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

See Also
- setDynamicDepthLimit: (page 111)

Declared In
NSWindow.h

hasShadow

Indicates whether the window has a shadow.

```
- (BOOL)hasShadow
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Return Value
YES when the window has a shadow; NO otherwise.

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

See Also
- setHasShadow: (page 117)
- invalidateShadow (page 71)

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 117)

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; NO otherwise.

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

See Also
- setIgnoresMouseEvents: (page 118)

Declared In
NSWindow.h

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 118)
- `setNextKeyView:` (`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. Either it can be `NSBorderlessWindowMask`, or it can contain any of the options described in "Constants" (page 151), 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 "Constants" (page 151).

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 108) method.

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

See Also
- `orderFront:` (page 85)
- `setTitle:` (page 127)
- `setOneShot:` (page 122)
- `initWithContentRect:styleMask:backing:defer:screen:` (page 69)

Related Sample Code
CIAnnotation
FancyAbout
FunkyOverlayWindow
TrackBall
UIElementInspector

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 "Constants" (page 151), 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 "Constants" (page 151).

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 85)
- `setTitle:` (page 127)
- `setOneShot:` (page 122)

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 137)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Related Sample Code
CarbonInCocoa

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 93)
- resetCursorRects (NSView)

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 66)
- setHasShadow: (page 117)

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; NO otherwise.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 101)
- `displayIfNeeded` (page 57)
- `setNeedsDisplay:` (`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; `NO` otherwise.

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; `NO` otherwise.

Discussion
The default initial setting is `NO`.

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

See Also
- `setExcludedFromWindowsMenu:` (page 112)

Declared In
`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

isFlushWindowDisabled

Indicates whether the window's flushing ability is disabled.

```
- (BOOL)isFlushWindowDisabled
```

Return Value
`YES` when the window's flushing ability has been disabled; `NO` otherwise.

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

See Also
- `disableFlushWindow` (page 55)
- `enableFlushWindow` (page 60)

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; `NO` otherwise.

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

See Also
- `isMainWindow` (page 73)
- `makeKeyWindow` (page 80)

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, `NO` otherwise.

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

See Also
- `isKeyWindow` (page 73)
- `makeMainWindow` (page 80)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 81)

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; NO otherwise.

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 121)

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
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Return Value
`YES` when the window's window device is freed when it's removed from the screen list; `NO` otherwise.

Discussion
The default is `NO`.

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

See Also
`- setOneShot:` (page 122)

Declared In
`NSWindow.h`

isOpaque

Indicates whether the window is opaque.

```
- (BOOL)isOpaque
```

Return Value
`YES` when the window is opaque; `NO` otherwise.

Discussion
The default is `YES`.

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

See Also
`- setOpaque:` (page 122)

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 124)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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); NO otherwise.

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

See Also
- visibleRect (NSView)

Related Sample Code
GLChildWindowDemo
SimpleCocoaJavaMovie
SimpleCocoaJavaMovieCocoa

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 150), 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: (page 150), a default frame that nearly fits the screen is chosen. If the delegate or window class implements windowWillResize:toSize: (page 148), 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 138)

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 98)
- `nextKeyView` (`NSView`)
- `performMnemonic:` (`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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Discussion
This direction can be one of the values described in "Constants" (page 151).

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

See Also
- `selectNextKeyView:` (page 98)
- `selectPreviousKeyView:` (page 99)

Declared In
`NSWindow.h`

level

Returns the window level of the window.

```
- (NSInteger)level
```

Return Value
The window level.

Discussion
See "Constants" (page 151) for a list of possible values.

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

See Also
- `setLevel:` (page 118)

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; `NO` otherwise.

Discussion
If *responder* isn't already the first responder, this method first sends a `resignFirstResponder` 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

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

`becomeFirstResponder` 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` 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` to *`responder`*, and do not call `makeFirstResponder:` if `acceptsFirstResponder` returns `NO`.

Use `setInitialFirstResponder:` (page 118) 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` (`NSResponder`)
- `resignFirstResponder` (`NSResponder`)

Related Sample Code
Aperture Edit Plugin - Borders & Titles
CustomSave
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 85)
- `orderBack:` (page 84)
- `orderOut:` (page 86)
- `orderWindow:relativeTo:` (page 86)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- setLevel: (page 118)

Related Sample Code
GridCalendar
QTAudioExtractionPanel
Quartz Composer WWDC 2005 TextEdit
TextEditPlus
WhackedTV

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 80)
- becomeKeyWindow (page 40)
- isKeyWindow (page 73)

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 80)
- becomeMainWindow (page 40)
- isMainWindow (page 73)

Declared In
NSWindow.h

maxSize

Returns the maximum size to which the window's frame (including its title bar) can be sized.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
- (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 112).

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

See Also
- `setMaxSize:` (page 119)
- `minSize` (page 82)
- `aspectRatio` (page 37)
- `resizeIncrements` (page 95)

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 53)

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

See Also
- `setMiniwindowImage:` (page 120)
- `miniwindowTitle` (page 82)

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 120)
- `miniwindowImage` (page 81)

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 112).

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

See Also
- `setMinSize:` (page 121)
- `maxSize` (page 80)
- `aspectRatio` (page 37)
- `resizeIncrements` (page 95)

Declared In
`NSWindow.h`

mouseLocationOutsideOfEventStream

Returns the current location of the pointer reckoned in the window's base coordinate system.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
- (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`.

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

See Also
`- currentEvent` (`NSApplication`)

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:` 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:` (`NSApplication`)

Related Sample Code
Sketch-112

Declared In
`NSWindow.h`

nextEventMatchingMask:untilDate:inMode:dequeue:

Forwards the message to the global `NSApplication` object, `NSApp`.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
- (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:` (`NSApplication`)

Related Sample Code

CIAnnotation

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 85)
- `orderOut:` (page 86)
- `orderWindow:relativeTo:` (page 86)
- `makeKeyAndOrderFront:` (page 79)
- `level` (page 78)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 84)
- orderOut: (page 86)
- orderWindow:relativeTo: (page 86)
- makeKeyAndOrderFront: (page 79)
- level (page 78)

Related Sample Code
Aperture Edit Plugin - Borders & Titles
CocoaDVDPlayer
FunkyOverlayWindow
TrackBall
UIElementInspector

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

See Also
- `orderFront:` (page 85)
- `level` (page 78)

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 86) method causes the window to be removed from the screen, but does not cause it to be released. See the `close` (page 45) method for information on when a window is released.

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

See Also
- `orderFront:` (page 85)
- `orderBack:` (page 84)
- `orderWindow:relativeTo:` (page 86)
- `setReleasedWhenClosed:` (page 124)

Related Sample Code
EnhancedDataBurn
GridCalendar
ImageClient
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

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
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Parameters

orderingMode

NSWindowOut: The window is removed from the screen list and *otherWindowNumber* is ignored.

NSWindowAbove: The window is ordered immediately in front of the window whose window number is *otherWindowNumber*

NSWindowBelow: 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 85)
- orderBack: (page 84)
- orderOut: (page 86)
- makeKeyAndOrderFront: (page 79)
- level (page 78)
- windowNumber (page 137)

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 92)
- childWindows (page 44)
- addChildWindow:ordered: (page 34)
- setParentWindow: (page 123)

Related Sample Code
FunkyOverlayWindow
GLChildWindowDemo

Declared In
NSWindow.h

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 146), 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 146) method returns `NO`, the window isn't closed. If it returns `YES`, or if it isn't implemented, `performClose:` (page 88) invokes the `close` (page 45) 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 146) message), the system emits the alert sound.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `styleMask` (page 132)
- `performMiniaturize:` (page 88)

Related Sample Code

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 45)
- `styleMask` (page 132)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- performClose: (page 88)

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 132)
- zoom: (page 138)

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:

Declared In
NSWindow.h

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 151) for possible values.

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

See Also
- `setPreferredBackingLocation:` (page 123)
- `backingLocation` (page 39)

Declared In
`NSWindow.h`

preservesContentDuringLiveResize

Returns a Boolean value that indicates whether the window tries to optimize live 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; `NO` otherwise.

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` 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 124)
- `preservesContentDuringLiveResize` (`NSView`)

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
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 103) to have the window recalculate the loop automatically.

Availability

Available in Mac OS X v10.4 and later.

See Also

- selectKeyViewFollowingView: (page 97)
- selectKeyViewPrecedingView: (page 98)
- setAutorecalculatesKeyViewLoop: (page 103)

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

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

delegate if the delegate responds to the selector of the message. The messages forwarded this way are `draggingEntered:`, `draggingUpdated:`, `draggingExited:`, `prepareForDragOperation:`, `performDragOperation:`, and `concludeDragOperation:`.

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

See Also
- `unregisterDraggedTypes` (page 134)

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 34)
- `childWindows` (page 44)
- `parentWindow` (page 87)
- `setParentWindow:` (page 123)

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 125)

Declared In
`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 1.

Table 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 140).

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

See Also
- `setRepresentedURL:` (page 125)
- `window:shouldDragDocumentWithEvent:from:withPasteboard:` (page 139)

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 56) to clear the window's cursor rectangles, then sends `resetCursorRects` (page 93) 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 71).

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 94) to the window's first responder, sends `windowDidResignKey:` (page 145) to the window's delegate, and posts an `NSWindowDidResignKeyNotification` (page 164) to the default notification center.

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

See Also
- `becomeKeyWindow` (page 40)
- `resignMainWindow` (page 94)

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 145) to the window's delegate and posts an `NSWindowDidResignMainNotification` (page 164) to the default notification center.

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

See Also
- `becomeMainWindow` (page 40)
- `resignKeyWindow` (page 94)

Declared In
NSWindow.h

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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` 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 148).

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 126)
- `setAspectRatio:` (page 101)
- `setFrame:display:` (page 112)

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 40).

```
- (void)restoreCachedImage
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Discussion
You must invoke `flushWindow` (page 63) 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 56)
- `display` (page 57)

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 116), 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 131)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 51)

Related Sample Code
CocoaDVDPlayer
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` message to *referenceView* and, if that message returns an `NSView` object, invokes `makeFirstResponder:` (page 78) with the returned object.

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

See Also
- `selectKeyViewPrecedingView:` (page 98)

Declared In
NSWindow.h

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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` message to *referenceView* and, if that message returns an `NSView` object, invokes `makeFirstResponder:` (page 78) with the returned object.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- selectKeyViewFollowingView:` (page 97)

Declared In

`NSWindow.h`

selectNextKeyView:

This action method searches for a candidate next key view and, if it finds one, invokes `makeFirstResponder:` (page 78) 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` method of `NSView`
- The object designated as the window's initial first responder (using `setInitialFirstResponder:` (page 118)) if it returns `YES` to an `acceptsFirstResponder` 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 99)

`- selectKeyViewFollowingView:` (page 97)

Declared In

`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

selectPreviousKeyView:

This action method searches for a candidate previous key view and, if it finds one, invokes `makeFirstResponder:` (page 78) 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` method of `NSView`
- The object designated as the window's initial first responder (using `setInitialFirstResponder:` (page 118)) if it returns `YES` to an `acceptsFirstResponder` 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 98)
- `selectKeyViewPrecedingView:` (page 98)

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:` message with a window number of 0.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 34)

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 35)

Declared In
NSWindow.h

setAlphaValue:

Applies a given alpha value to the entire window.

```
- (void)setAlphaValue:(CGFloat)windowAlpha
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Parameters

windowAlpha

The alpha value to apply.

Availability

Available in Mac OS X v10.0 and later.

See Also

- alphaValue (page 36)

Related Sample Code

FunkyOverlayWindow

JavaSplashScreen

RoundTransparentWindow

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 126) message with the width and height set to 1.0:

```
[myWindow setResizeIncrements:NSMakeSize(1.0,1.0)];
```

The setContentAspectRatio: (page 105) method takes precedence over this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- aspectRatio (page 37)
- setFrame:display: (page 112)

Declared In

NSWindow.h

setAutodisplay:

Specifies whether the window is to automatically display the views that are marked as needing it.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
- (void)setAutodisplay:(BOOL)autodisplay
```

Parameters

autodisplay

`YES` to have the window automatically display views that need to be displayed; `NO` to specify otherwise.

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 71)
- `displayIfNeeded` (page 57)
- `displayIfNeeded` (`NSView`)

Declared In

`NSWindow.h`

setAutorecalculatesContentBorderThickness:forEdge:

Specifies whether the window calculates the thickness of a given border automatically.

```
- (void)setAutorecalculatesContentBorderThickness:(BOOL)autorecalculateContentBorderThickness
    forEdge:(NSRectEdge)edge
```

Parameters

autorecalculateContentBorderThickness

`YES` to have the window calculate the thickness of edge automatically; `NO` otherwise.

edge

Border whose thickness autorecalculation status to set:

- `NSMaxYEdge`: Top border.
- `NSMinYEdge`: Bottom border.

Special Considerations

Turning off a border's autorecalculation status sets its border thickness to `0.0`.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `autorecalculatesContentBorderThicknessForEdge:` (page 38)
- `contentBorderThicknessForEdge:` (page 47)

Declared In

`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

setAutorecalculatesKeyViewLoop:

Specifies whether to recalculate the key view loop automatically when views are added or removed.

```
- (void)setAutorecalculatesKeyViewLoop:(BOOL)autorecalculateKeyViewLoop
```

Parameters

autorecalculateKeyViewLoop

YES to recalculate the key view loop automatically; NO otherwise.

Discussion

If *autorecalculateKeyViewLoop* is NO, the client code must update the key view loop manually or call recalculateKeyViewLoop (page 91) to have the window recalculate it.

Availability

Available in Mac OS X v10.4 and later.

See Also

- autorecalculatesKeyViewLoop (page 38)
- recalculateKeyViewLoop (page 91)

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 39)

Related Sample Code

JavaSplashScreen

RoundTransparentWindow

UIElementInspector

Declared In

NSWindow.h

setBackingType:

Sets the window's backing store type to a given type.

```
- (void)setBackingType:(NSBackingStoreType)backingType
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Parameters

backingType

The backing store type to set.

Discussion

The valid backing store types are described in "Constants" (page 151).

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 39)
- `initWithContentRect:styleMask:backing:defer:` (page 68)
- `initWithContentRect:styleMask:backing:defer:screen:` (page 69)

Declared In

`NSWindow.h`

setCanBecomeVisibleWithoutLogin:

Specifies whether the window can be displayed at the login window.

```
- (void)setCanBecomeVisibleWithoutLogin:(BOOL)canBecomeVisibleWithoutLogin
```

Parameters

canBecomeVisibleWithoutLogin

`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 42)

Declared In

`NSWindow.h`

setCanHide:

Specifies whether the window can be hidden when its application becomes hidden (during execution of the `NSApplication hide:` method).

```
- (void)setCanHide:(BOOL)canHide
```

Parameters

canHide

`YES` specifies that the window can be hidden when its application becomes hidden; `NO` specifies otherwise.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

See Also
- `canHide` (page 42)

Declared In
`NSWindow.h`

setCollectionBehavior:

Specifies the window's behavior in window collections.

```
- (void)setCollectionBehavior:(NSWindowCollectionBehavior)collectionBehavior;
```

Parameters

collectionBehavior
The collection behavior identifier to set.

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

See Also
- `collectionBehavior` (page 45)

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 101).

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

See Also
- `contentAspectRatio` (page 46)

Declared In
`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

setContentBorderThickness:forEdge:

Specifies the thickness of a given border of the window.

```
- (void)setContentBorderThickness:(CGFloat)borderThickness forEdge:(NSRectEdge)edge
```

Parameters

borderThickness

Thickness for *edge*, in points.

edge

Border whose thickness to set:

- `NSMaxYEdge`: Top border.
- `NSMinYEdge`: Bottom border.

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

See Also
- `contentBorderThicknessForEdge:` (page 47)

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 107) method and the `setFrame...` methods other than `setFrame:display:` (page 112). This method takes precedence over `setMaxSize:` (page 119).

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

See Also
- `contentMaxSize` (page 47)
- `setContentMinSize:` (page 106)

Declared In
`NSWindow.h`

setContentMinSize:

Sets the minimum size of the window's content view in the window's base coordinate system.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
- (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 107) method and the `setFrame...` methods other than `setFrame:display:` (page 112). This method takes precedence over `setMinSize:` (page 121).

Availability

Available in Mac OS X v10.3 and later.

See Also

- `contentMinSize` (page 48)
- `setContentMaxSize:` (page 106)

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 resizer 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 126).

Availability

Available in Mac OS X v10.3 and later.

See Also

- `contentResizeIncrements` (page 48)

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
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 112)

`+ contentRectForFrameRect:styleMask:` (page 31)

`+ frameRectForContentRect:styleMask:` (page 32)

Related Sample Code

CocoaDVDPlayer

CocoaVideoFrameToGWorld

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

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 49)

`- setContentSize:` (page 107)

Related Sample Code

CustomSave

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

FunkyOverlayWindow
GLChildWindowDemo

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 52)
- disableKeyEquivalentForDefaultButtonCell (page 55)
- enableKeyEquivalentForDefaultButtonCell (page 60)

Declared In
NSWindow.h

setDelegate:

Sets the window's delegate to a given object or removes an existing delegate.

```
- (void)setDelegate:(id)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 52)
- tryToPerform:with: (page 133)
- sendAction:to:from: (NSApplication)

Related Sample Code
AudioBurn

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

DataBurn
Quartz Composer WWDC 2005 TextEdit
TextEditPlus
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 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.

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

See Also
- depthLimit (page 53)
+ defaultDepthLimit (page 31)
- setDynamicDepthLimit: (page 111)

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 162) notification.

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

See Also
`- displaysWhenScreenProfileChanges` (page 58)

Declared In
`NSWindow.h`

setDocumentEdited:

Specifies whether the window's document has been edited.

```
- (void)setDocumentEdited:(BOOL)documentEdited
```

Parameters
documentEdited
`YES` to specify that the window's document has been edited; `NO` to specify otherwise.

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 72) to determine whether to allow the user a chance to save the document.

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

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
`YES` specifies a dynamic depth limit; `NO` specifies otherwise.

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 110) method.

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

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

See Also
- `hasDynamicDepthLimit` (page 66)
+ `defaultDepthLimit` (page 31)

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

`YES` to specify that the window is to be omitted from the application's Windows menu; `NO` to specify otherwise.

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

See Also
- `isExcludedFromWindowsMenu` (page 72)

Related Sample Code
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.

displayViews

Specifies whether the window redraws the views that need to be displayed. When `YES` the window sends a `displayIfNeeded` (page 57) 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 64)
- `setFrameFromString:` (page 114)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

- `setFrameOrigin:` (page 115)
- `setFrameTopLeftPoint:` (page 115)
- `setFrameUsingName:` (page 116)

Related Sample Code
ColorMatching
FunkyOverlayWindow
SimpleCocoaMovie
SimpleCocoaMovieQT

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.

displayViews
Specifies whether the window redraws the views that need to be displayed. When `YES` the window sends a `displayIfNeeded` (page 57) 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 36).

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

Related Sample Code
iSpend
QTQuartzPlayer

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
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 96)) 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 33)
- stringWithSavedFrame (page 131)
- setFrameFromString: (page 114)

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 131).

Discussion

The frame is constrained according to the window's minimum and maximum size settings. This method causes a windowWillResize:toSize: (page 148) message to be sent to the delegate.

Availability

Available in Mac OS X v10.0 and later.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 112)
- setFrameTopLeftPoint: (page 115)

Related Sample Code
FunkyOverlayWindow

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 43)
- setFrame:display: (page 112)
- setFrameOrigin: (page 115)

Related Sample Code
CocoaDVDPlayer

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Quartz Composer WWDC 2005 TextEdit
TextEditPlus

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; NO otherwise.

Discussion
The frame is constrained according to the window's minimum and maximum size settings. This method causes a windowWillResize:toSize: (page 148) message to be sent to the delegate.

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

See Also
- setFrameAutosaveName: (page 113)
+ removeFrameUsingName: (page 33)
- stringWithSavedFrame (page 131)
- setFrameFromString: (page 114)

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 116) on a nonresizable window; NO to fail on a nonresizable window.

Return Value
YES when *frameName* is read and the frame is set successfully; NO otherwise.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

YES specifies that the window has a shadow; NO specifies otherwise.

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 66)
- invalidateShadow (page 71)

Related Sample Code
FunkyOverlayWindow
JavaSplashScreen
RoundTransparentWindow

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 67)

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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
YES to have the window ignore mouse events; NO to specify otherwise.

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

See Also
- ignoresMouseEvents (page 67)

Related Sample Code
FunkyOverlayWindow

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 68)

Declared In
NSWindow.h

setLevel:

Sets the window's window level to a given level.

```
- (void)setLevel:(NSInteger)windowLevel
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Parameters

windowLevel

The window level to set.

Discussion

Some useful predefined values, ordered from lowest to highest, are described in "Constants" (page 151).

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 78)
- `orderWindow:relativeTo:` (page 86)
- `orderFront:` (page 85)
- `orderBack:` (page 84)

Related Sample Code

FunkyOverlayWindow

JavaSplashScreen

RoundTransparentWindow

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 112). 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 106) method takes precedence over this method.

Availability

Available in Mac OS X v10.0 and later.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

See Also
- `maxSize` (page 80)
- `setMinSize:` (page 121)
- `setAspectRatio:` (page 101)
- `setResizeIncrements:` (page 126)

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 165). 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 81)
- `isMiniaturized` (page 74)

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
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 127) or `setTitleWithRepresentedFilename:` (page 128)) automatically changes the minimized window's title as well.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `miniwindowTitle` (page 82)

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 112).

The `setContentMinSize:` (page 106) method takes precedence over this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `minSize` (page 82)
- `setMaxSize:` (page 119)
- `setAspectRatio:` (page 101)
- `setResizeIncrements:` (page 126)

Declared In

`NSWindow.h`

setMovableByWindowBackground:

Sets whether the window is movable by clicking and dragging anywhere in its background.

```
- (void)setMovableByWindowBackground:(BOOL)movableByWindowBackground
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 74)

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 74)

Related Sample Code
VideoViewer

Declared In
NSWindow.h

setOpaque:

Specifies whether the window is opaque.

```
- (void)setOpaque:(BOOL)opaque
```

Parameters

opaque

YES specifies that the window is opaque; NO specifies otherwise.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

See Also
- `isOpaque` (page 75)

Related Sample Code
FunkyOverlayWindow
JavaSplashScreen
RoundTransparentWindow
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
You should call super if overriding.

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

See Also
- `removeChildWindow:` (page 92)
- `childWindows` (page 44)
- `parentWindow` (page 87)
- `addChildWindow:ordered:` (page 34)

Declared In
`NSWindow.h`

setPreferredBackingLocation:

Specifies the preferred location for the window's backing store.

```
- (void)setPreferredBackingLocation:(NSWindowBackingLocation)preferredBackingLocation
```

Parameters
preferredBackingLocation
The preferred location for the window's backing store. See "Constants" (page 151) for possible values.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Discussion
Use only when optimizing for performance.

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

See Also
- `preferredBackingLocation` (page 90)

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 90)

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 146) message.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

See Also
- `close` (page 45)
- `isReleasedWhenClosed` (page 75)

Related Sample Code
Fiendishthngs
UIElementInspector
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 92)
- `setTitleWithRepresentedFilename:` (page 128)

Declared In
`NSWindow.h`

setRepresentedURL:

Specifies the URL of the file the window represents.

```
- (void)setRepresentedURL:(NSURL *)representedURL
```

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 93)

Declared In
`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 resizer 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 101).

The setContentResizeIncrements: (page 107) method takes precedence over this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- resizeIncrements (page 95)
- setFrame:display: (page 112)

Declared In

NSWindow.h

setSharingType:

Specifies the level of access other processes have to the window's content.

```
- (void)setSharingType:(NSWindowSharingType)sharingType
```

Parameters

sharingType

The sharing level of the window's content. See "Constants" (page 151) for possible values.

Availability

Available in Mac OS X v10.5 and later.

See Also

- sharingType (page 130)

Declared In

NSWindow.h

setShowsResizeIndicator:

Specifies whether the window's resize indicator is visible

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
- (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 130)

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 130)

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

See Also
- `title` (page 132)
- `setTitleWithRepresentedFilename:` (page 128)
- `setMiniwindowTitle:` (page 120)

Related Sample Code
QTCoreVideo102
QTCoreVideo201
QTCoreVideo301
VertexPerformanceTest
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 125).

```
- (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.

This method also sets the title bar of the window's minimized window.

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

See Also
- `title` (page 132)
- `setTitle:` (page 127)
- `setMiniwindowTitle:` (page 120)

Declared In
`NSWindow.h`

setToolbar:

Sets the window's toolbar.

```
- (void)setToolbar:(NSToolbar *)toolbar
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 133)

Related Sample Code

iSpend

PDFKitLinker2

Declared In

`NSWindow.h`

setViewsNeedDisplay:

Specifies whether the window's views need to be displayed..

```
- (void)setViewsNeedDisplay:(BOOL)viewsNeedDisplay
```

Parameters

viewsNeedDisplay

`YES` to specify that the window's views need to be displayed; `NO` to specify otherwise.

Discussion

You should rarely need to invoke this method; the `NSView` method `setNeedsDisplay:` and similar methods invoke it automatically.

Availability

Available in Mac OS X v10.0 and later.

See Also

- viewsNeedDisplay (page 136)

Declared In

`NSWindow.h`

setWindowController:

Sets the window's window controller.

```
- (void)setWindowController:(NSWindowController *)windowController
```

Parameters

windowController

Window controller to set.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

See Also
- `windowController` (page 136)

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 "Constants" (page 151) for possible values.

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

See Also
- `setSharingType:` (page 126)

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 126)

Declared In
`NSWindow.h`

showsToolbarButton

Indicates whether the toolbar control button is currently displayed.

```
- (BOOL)showsToolbarButton
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Return Value
`YES` if the standard toolbar button is currently displayed; `NO` otherwise.

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 127)

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 33)

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 114) message.

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

Declared In
`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 "Constants" (page 151). A window's style is set when the object is initialized. Once set, it can't be changed.

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

Related Sample Code
CocoaDragAndDrop
GLChildWindowDemo
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

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 128), this method returns the file's path.

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

See Also
`- setTitle:` (page 127)

Related Sample Code
UIElementInspector

Declared In
NSWindow.h

toggleToolbarShown:

The action method for the "Hide Toolbar" menu item (which alternates with "Show Toolbar").

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
- (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 128)

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*; *NO* otherwise.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Discussion
The window tries to perform the method *selector* using its inherited `NSResponder` method `tryToPerform:with:`. 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 91)

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 165) 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` method.

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

See Also
`- setWindowsNeedUpdate:` (`NSApplication`)

Declared In
`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

useOptimizedDrawing:

Specifies whether the window is to optimize focusing and drawing when displaying its views.

```
- (void)useOptimizedDrawing:(BOOL)optimizedDrawing
```

Parameters

optimizedDrawing

YES to have the window optimize focusing and drawing for its views; NO to specify otherwise, in which case, the window does not preserve the Z-ordering of overlapping views when an object explicitly sends lockFocus 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.

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.

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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:` (`NSResponder`)
- `validRequestorForSendType:returnType:` (`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; `NO` otherwise.

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

See Also
- `setViewsNeedDisplay:` (page 129)

Declared In
`NSWindow.h`

windowController

Returns the window's window controller.

```
- (id)windowController
```

Return Value
The window's window controller.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

See Also
`- setWindowController:` (page 129)

Related Sample Code
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 86) method and in the Application Kit function `NSWindowList`..

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 68)
`- setOneShot:` (page 122)

Related Sample Code
CocoaDVDPlayer
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
`NSWindow.h`

windowRef

Returns the Carbon `WindowRef` associated with the window, creating one if necessary.

```
- (void *)windowRef
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 70)

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; `NO` otherwise.

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:(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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Discussion
For more information on the standard and user states, see `windowWillUseStandardFrame:defaultFrame:` (page 150).

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 89) method. It performs the following steps:

1. Invokes the `windowWillUseStandardFrame:defaultFrame:` (page 150) 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 147) 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 76)

Declared In
`NSWindow.h`

Delegate Methods

window:shouldDragDocumentWithEvent:from:withPasteboard:

Determines whether the sender allows the user to drag the sender's represented file's icon from the sender's title bar.

```
- (BOOL)window:(NSWindow *)sender shouldDragDocumentWithEvent:(NSEvent *)mouseEvent
    from:(NSPoint)startPoint withPasteboard:(NSPasteboard *)pasteboard
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Parameters

sender

The window whose represented file's icon the user wants to drag.

mouseEvent

The left-mouse down event that triggered the dragging operation.

startPoint

The location at which the user started the dragging operation.

pasteboard

The pasteboard containing the contents of the represented file, 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.

See Also

- representedURL (page 93)

Declared In

`NSWindow.h`

window:shouldPopUpDocumentPathMenu:

Determines whether the sender displays the title pop-up menu in response to a Command-click on the sender's title.

```
- (BOOL)window:(NSWindow *)sender shouldPopUpDocumentPathMenu:(NSMenu *)titleMenu
```

Parameters

sender

The window whose title the user Command-clicked.

titleMenu

The menu the sender displays, if allowed. By default its items are the path components of the file represented by *sender*.

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.

See Also

- representedURL (page 93)

Declared In

`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

window:willPositionSheet:usingRect:

Sent to the delegate just before the animation of a sheet, giving it the opportunity to return a custom location for the attachment of a sheet to a window.

```
- (NSRect)window:(NSWindow *)window willPositionSheet:(NSWindow *)sheet
    usingRect:(NSRect)defaultSheetRect
```

Parameters

window

The window containing the sheet to be animated.

sheet

The sheet to be animated.

defaultSheetRect

The default sheet location, just under the title bar of the window, aligned with the left and right edges of the window.

Return Value

A custom location for the attachment of *sheet* to *window*.

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 *defaultSheetRect* 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 zero. 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.

Declared In

`NSWindow.h`

windowDidBecomeKey:

Sent by the default notification center immediately after an `NSWindow` object has become key.

```
- (void)windowDidBecomeKey:(NSNotification *)notification
```

Parameters

notification

`NSWindowDidBecomeKeyNotification` (page 161).

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Discussion
You can retrieve the `NSWindow` object in question by sending `object` to *`notification`*.

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

Declared In
`NSWindow.h`

windowDidBecomeMain:

Sent by the default notification center immediately after an `NSWindow` object has become main.

```
- (void)windowDidBecomeMain:(NSNotification *)notification
```

Parameters
`notification`
`NSWindowDidBecomeMainNotification` (page 162).

Discussion
You can retrieve the `NSWindow` object in question by sending `object` to *`notification`*.

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

Declared In
`NSWindow.h`

windowDidChangeScreen:

Sent by the default notification center immediately after an `NSWindow` object has changed screens.

```
- (void)windowDidChangeScreen:(NSNotification *)notification
```

Parameters
`notification`
`NSWindowDidChangeScreenNotification` (page 162).

Discussion
You can retrieve the `NSWindow` object in question by sending `object` to *`notification`*.

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

Declared In
`NSWindow.h`

windowDidChangeScreenProfile:

Sent by the default notification center immediately after an `NSWindow` object has changed screen display profiles.

```
- (void)windowDidChangeScreenProfile:(NSNotification *)notification
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Parameters

notification

`NSWindowDidChangeScreenProfileNotification` (page 162).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSWindow.h`

windowDidDeminiaturize:

Sent by the default notification center immediately after an `NSWindow` object has been deminimized.

```
- (void)windowDidDeminiaturize:(NSNotification *)notification
```

Parameters

notification

`NSWindowDidDeminiaturizeNotification` (page 163).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSWindow.h`

windowDidEndSheet:

Sent by the default notification center immediately after an `NSWindow` object closes a sheet.

```
- (void)windowDidEndSheet:(NSNotification *)notification
```

Parameters

notification

`NSWindowDidEndSheetNotification` (page 163).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

windowDidExpose:

Sent by the default notification center immediately after an `NSWindow` object has been exposed.

```
- (void)windowDidExpose:(NSNotification *)notification
```

Parameters

notification

`NSWindowDidExposeNotification` (page 163).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSWindow.h`

windowDidMiniaturize:

Sent by the default notification center immediately after an `NSWindow` object has been minimized.

```
- (void)windowDidMiniaturize:(NSNotification *)notification
```

Parameters

notification

`NSWindowDidMiniaturizeNotification` (page 163).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSWindow.h`

windowDidMove:

Sent by the default notification center immediately after an `NSWindow` object has been moved.

```
- (void)windowDidMove:(NSNotification *)notification
```

Parameters

notification

`NSWindowDidMoveNotification` (page 164).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.0 and later.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Declared In
NSWindow.h

windowDidResignKey:

Sent by the default notification center immediately after an NSWindow object has resigned its status as key window.

```
- (void)windowDidResignKey:(NSNotification *)notification
```

Parameters
notification
NSWindowDidResignKeyNotification (page 164).

Discussion
You can retrieve the NSWindow object in question by sending object to *notification*.

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

Declared In
NSWindow.h

windowDidResignMain:

Sent by the default notification center immediately after an NSWindow object has resigned its status as main window.

```
- (void)windowDidResignMain:(NSNotification *)notification
```

Parameters
notification
NSWindowDidResignMainNotification (page 164).

Discussion
You can retrieve the NSWindow object in question by sending object to *notification*.

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

Declared In
NSWindow.h

windowDidResize:

Sent by the default notification center immediately after a window has been resized.

```
- (void)windowDidResize:(NSNotification *)notification
```

Parameters
notification
NSWindowDidResizeNotification (page 164).

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Discussion
You can retrieve the `NSWindow` object in question by sending `object` to *`notification`*.

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

Declared In
`NSWindow.h`

windowDidUpdate:

Sent by the default notification center immediately after an `NSWindow` object receives an `update` (page 134) message.

```
- (void)windowDidUpdate:(NSNotification *)notification
```

Parameters
`notification`
`NSWindowDidUpdateNotification` (page 165).

Discussion
You can retrieve the `NSWindow` object in question by sending `object` to *`notification`*.

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

Declared In
`NSWindow.h`

windowShouldClose:

Invoked when the user attempts to close a window or a window receives a `performClose:` (page 88) message.

```
- (BOOL)windowShouldClose:(id)window
```

Parameters
`window`
The window being closed.

Return Value
`YES` to allow *`window`* 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.

Declared In
`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

windowShouldZoom:toFrame:

Sent just before *sender* is zoomed to allow or disallow the operation.

```
- (BOOL)windowShouldZoom:(NSWindow *)window toFrame:(NSRect)proposedFrame
```

Parameters

window

The window being zoomed.

proposedFrame

The rectangle to which *window* is being zoomed.

Return Value

`YES` to allow the *window* frame to become *proposedFrame*; `NO` otherwise.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- windowWillUseStandardFrame:defaultFrame:` (page 150)

Declared In

`NSWindow.h`

windowWillBeginSheet:

Sent by the default notification center immediately before an `NSWindow` object opens a sheet.

```
- (void)windowWillBeginSheet:(NSNotification *)notification
```

Parameters

notification

`NSWindowWillBeginSheetNotification` (page 165).

Discussion

You can retrieve the `NSWindow` object in question by sending `object` to *notification*.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`NSWindow.h`

windowWillClose:

Sent by the default notification center immediately before an `NSWindow` object closes.

```
- (void)windowWillClose:(NSNotification *)notification
```

Parameters

notification

`NSWindowWillCloseNotification` (page 165).

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Discussion
You can retrieve the `NSWindow` object in question by sending `object` to *`notification`*.

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

Declared In
`NSWindow.h`

windowWillMiniaturize:

Sent by the default notification center immediately before an `NSWindow` object is minimized.

```
- (void)windowWillMiniaturize:(NSNotification *)notification
```

Parameters
`notification`
`NSWindowWillMiniaturizeNotification` (page 165).

Discussion
You can retrieve the `NSWindow` object in question by sending `object` to *`notification`*.

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

Declared In
`NSWindow.h`

windowWillMove:

Sent by the default notification center immediately before an `NSWindow` object is moved.

```
- (void)windowWillMove:(NSNotification *)notification
```

Parameters
`notification`
`NSWindowWillMoveNotification` (page 166).

Discussion
You can retrieve the `NSWindow` object in question by sending `object` to *`notification`*.

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

Declared In
`NSWindow.h`

windowWillResize:toSize:

Invoked when a window is being resized (whether by the user or through one of the `setFrame...` methods other than `setFrame:display:` (page 112)).

```
- (NSSize)windowWillResize:(NSWindow *)window toSize:(NSSize)proposedFrameSize
```


2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Parameters

window

The window being resized.

proposedFrameSize

The size to which *window* is being resized.

Discussion

The *proposedFrameSize* contains the size (in screen coordinates) the 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. The `NSWindow` object'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.

Declared In

`NSWindow.h`

windowWillReturnFieldEditor:toObject:

Invoked when the field editor for a text-displaying object is requested.

```
- (id)windowWillReturnFieldEditor:(NSWindow *)window toObject:(id)anObject
```

Parameters

window

The window that is requesting the field editor from the delegate.

anObject

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 `anObject`; returns `nil` when the delegate has no field editor to assign.

Discussion

This method may be called multiple times while a control is first responder. Therefore, you must return the same field editor object for the control while the control is being edited.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- fieldEditor:forObject:` (page 61)

Declared In

`NSWindow.h`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

windowWillReturnUndoManager:

Invoked when the undo manager for a window is 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 *window*.

Discussion

If this method is not implemented by the delegate, the `NSWindow` object creates an `NSUndoManager` object for *window*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSWindow.h`

windowWillUseStandardFrame:defaultFrame:

Invoked by the `zoom:` (page 138) method while determining a frame an `NSWindow` object may be zoomed to.

```
- (NSRect)windowWillUseStandardFrame:(NSWindow *)window
    defaultFrame:(NSRect)defaultFrame
```

Parameters

window

The window whose frame size is being determined.

defaultFrame

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 standard frame for *window*.

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 138) method modifies the returned standard frame, if necessary, to fit on the current screen.

To customize the standard state, implement `windowWillUseStandardFrame:defaultFrame:` in the class of the window's delegate or, if necessary, in a window subclass. Your version should return a suitable standard frame, based on the currently displayed data or other factors.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

See Also
`- windowShouldZoom:toFrame:` (page 147)

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`.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

`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

These constants specify the window's level. 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 the `Window Level Keys` in Core Graphics.

```
#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
```

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`.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

`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`.

Declared In
`NSWindow.h`

Display Device—Descriptions

These constants are the keys for device description dictionaries used by `deviceDescription` (page 54).

```
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" 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`.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

`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`.

Declared In
`NSWindow.h`

Controlling the Look of a Window and Its Toolbar

This constant controls the look of a window and its toolbar.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
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`.

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 77).

```
typedef enum _NSSelectionDirection {
    NSDirectSelection = 0,
    NSSelectingNext,
    NSSelectingPrevious
} 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`.

Availability

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:

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
typedef enum {
     NSWindowCloseButton,
     NSWindowMiniaturizeButton,
     NSWindowZoomButton,
     NSWindowToolbarButton,
     NSWindowDocumentIconButton
} 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`.

Availability
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:`.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
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`.

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`, `NSBitsPerPixelFromDepth`, and `NSPlanarFromDepth` to extract info from an `NSWindowDepth` value. Use `NSBestDepth` to compute window depths. `NSBestDepth` 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`); other values maybe used as hints to provide backing stores of different configuration: for instance, 8-bit color.

Availability

Available in Mac OS X v10.0 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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
typedef enum _NSBackingStoreType {
    NSBackingStoreRetained     = 0,
    NSBackingStoreNonretained  = 1,
    NSBackingStoreBuffered     = 2
} 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`.

Availability

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 86).

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
typedef enum _NSWindowOrderingMode {
    NSWindowAbove         =  1,
    NSWindowBelow         = -1,
    NSWindowOut             =  0
} 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`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSGraphics.h`

NSWindowAuxiliaryOpaque

A private data structure used internally by `NSWindow`.

```
typedef struct NSWindowAuxiliary NSWindowAuxiliaryOpaque;
```

Availability

Available in Mac OS X v10.0 through Mac OS X v10.3.

Declared In

`NSWindow.h`

NSWindowSharingType

These constants and data type represent the access levels other processes can have to a window's content.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

```
typedef 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`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSWindow.h`

NSWindowBackingLocation

These constants and 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`.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

`NSWindowBackingLocationMainMemory`
Physical memory.

Available in Mac OS X v10.5 and later.

Declared in `NSWindow.h`.

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

Declared In
`NSWindow.h`

Managing Window Collections

These constants and data type identify window behavior in relation to window browsers and organizers, such as Spaces.

```
enum {
   NSWindowCollectionBehaviorDefault = 0,
   NSWindowCollectionBehaviorCanJoinAllSpaces = 1 << 0,
   NSWindowCollectionBehaviorMoveToActiveSpace = 1 << 1
};
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`.

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

Declared In
`NSWindow.h`

Notifications

NSWindowDidBecomeKeyNotification

Posted whenever an `NSWindow` object becomes the key window.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 58). 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`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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`

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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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 164) 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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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

Declared In
`NSWindow.h`

NSWindowDidUpdateNotification

Posted whenever an `NSWindow` object receives an `update` (page 134) message.

The notification object is the `NSWindow` object that received the `update` (page 134) 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.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

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`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Deprecated NSWindow Methods

A method identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.5

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.)

```
- (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`

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.)

```
- (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`

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Document Revision History

This table describes the changes to *NSWindow Class Reference*.

Date	Notes
2009-03-04	Fixed several typographical errors, added information to setFrameAutosaveName:, and documented several undocumented symbols.
2008-10-15	Enhanced discussion of NSBackingStoreType constants.
2007-10-31	Corrected typographical errors.
2007-09-04	Noted memory management policy exception for initWithWindowRef:.
2007-05-30	Updated for Mac OS X v10.5.
	Added usage details to `contentRectForFrameRect:styleMask:` (page 31) and `frameRectForContentRect:styleMask:` (page 32).
	Clarified and expanded descriptions of `firstResponder` (page 62), `windowRef` (page 137), and `fieldEditor:forObject:` (page 61).
	Clarified use of the `contentRectForFrameRect:styleMask:` (page 31) and `frameRectForContentRect:styleMask:` (page 32) methods.
	Reorganized “Tasks” section.
2006-11-07	Clarified descriptions of fieldEditor:forObject:, firstResponder, and windowRef.
2006-06-28	Made minor changes to adhere to reference consistency guidelines; enhanced the discussion of isZoomed.
2006-05-23	Clarified the return value of contentAspectRatio.
	First publication of this content as a separate document.

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Document Revision History

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

Index

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

E

F

G

H

I

K

L

M

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

N

O

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

P

R

S

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.

T

U

V

W

Z

2009-03-04 | © 2009 Apple Inc. All Rights Reserved.