Class

NSWindow

A window that an app displays on the screen.

macOS

```
@MainActor
class NSWindow
```

Overview

A single `NSWindow` object corresponds to, at most, one on-screen window. Windows perform two principal functions:

- To place views in a provided area
- To accept and distribute mouse and keyboard events the user generates to the appropriate views

> **Note**
>
> Although the `NSWindow` class inherits the `NSCoding` protocol from `NSResponder`, the class doesn't support coding. Legacy support for archivers exists, but its use is deprecated and may not work. Any attempt to archive or unarchive a window object using a keyed coding object raises an `invalidArgumentException` exception. For details about window restoration, see `restorationClass`.

Topics

Creating a Window

```
convenience init(contentViewController: NSViewController)
```

Creates a titled window that contains the specified content view controller.

```
init(contentRect: NSRect, styleMask: NSWindow.StyleMask, backing:
NSWindow.BackingStoreType, defer: Bool)
```

Initializes the window with the specified values.

```
convenience init(contentRect: NSRect, styleMask: NSWindow.StyleMask,
backing: NSWindow.BackingStoreType, defer: Bool, screen: NSScreen?)
```

Initializes an allocated window with the specified values.

Managing the Window’s Behavior

```
var delegate: (any NSWindowDelegate)?
```

The window’s delegate.

```
protocol NSWindowDelegate
```

A set of optional methods that a window’s delegate can implement to respond to events, such as window resizing, moving, exposing, and minimizing.

Configuring the Window’s Content

```
var contentViewController: NSViewController?
```

The main content view controller for the window.

```
var contentView: NSView?
```

The window’s content view, the highest accessible view object in the window’s view hierarchy.

Configuring the Window’s Appearance

```
var styleMask: NSWindow.StyleMask
```

Flags that describe the window’s current style, such as if it’s resizable or in full-screen mode.

```
struct StyleMask
```

Constants that specify the style of a window, and that you can combine with the C bitwise OR operator.

`func toggleFullScreen(Any?)`

Takes the window into or out of fullscreen mode,

`var worksWhenModal: Bool`

A Boolean value that indicates whether the window is able to receive keyboard and mouse events even when some other window is being run modally.

`var alphaValue: CGFloat`

The window's alpha value.

`var backgroundColor: NSColor!`

The color of the window's background.

`var colorSpace: NSColorSpace?`

The window's color space.

`func setDynamicDepthLimit(Bool)`

Sets a Boolean value that indicates whether the window's depth limit can change to match the depth of the screen it's on.

`var canHide: Bool`

A Boolean value that indicates whether the window can hide when its application becomes hidden.

`var isOnActiveSpace: Bool`

A Boolean value that indicates whether the window is on the currently active space.

`var hidesOnDeactivate: Bool`

A Boolean value that indicates whether the window is removed from the screen when its application becomes inactive.

`var collectionBehavior: NSWindow.CollectionBehavior`

A value that identifies the window's behavior in window collections.

`var isOpaque: Bool`

A Boolean value that indicates whether the window is opaque.

`var hasShadow: Bool`

A Boolean value that indicates whether the window has a shadow.

`func invalidateShadow()`

Invalidates the window shadow so that it is recomputed based on the current window shape.

```
func autorecalculatesContentBorderThickness(for: NSRectEdge) -> Bool
```

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

```
func setAutorecalculatesContentBorderThickness(Bool, for: NSRectEdge)
```

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

```
func contentBorderThickness(for: NSRectEdge) -> CGFloat
```

Indicates the thickness of a given border of the window.

```
func setContentBorderThickness(CGFloat, for: NSRectEdge)
```

Specifies the thickness of a given border of the window.

```
var preventsApplicationTerminationWhenModal: Bool
```

A Boolean value that indicates whether the window prevents application termination when modal.

```
var appearanceSource: (any NSAppearanceCustomization)!
```

An object that the window inherits its appearance from.

Accessing Window Information

```
var depthLimit: NSWindow.Depth
```

The depth limit of the window.

```
var hasDynamicDepthLimit: Bool
```

A Boolean value that indicates whether the window’s depth limit can change to match the depth of the screen it’s on.

```
class var defaultDepthLimit: NSWindow.Depth
```

Returns the default depth limit for instances of `NSWindow`.

```
var windowNumber: Int
```

The window number of the window’s window device.

```
class func windowNumbers(options: NSWindow.NumberListOptions) -> [
NSNumber]?
```

Returns the window numbers for all visible windows satisfying the specified options.

```
var deviceDescription: [NSDeviceDescriptionKey : Any]
```

A dictionary containing information about the window’s resolution, such as color, depth, and so on.

```
struct NSDeviceDescriptionKey
```

These constants are the keys for device description dictionaries.

```
var canBecomeVisibleWithoutLogin: Bool
```

A Boolean value that indicates whether the window can be displayed at the login window.

```
var sharingType: NSWindow.SharingType
```

A Boolean value that indicates the level of access other processes have to the window's content.

```
var backingType: NSWindow.BackingStoreType
```

The window's backing store type.

```
func displayLink(target: Any, selector: Selector) -> CADisplayLink
```

Getting Layout Information

```
class func contentRect(forFrameRect: NSRect, styleMask: NSWindow.Style
Mask) -> NSRect
```

Returns the content rectangle used by a window with a given frame rectangle and window style.

```
class func frameRect(forContentRect: NSRect, styleMask: NSWindow.Style
Mask) -> NSRect
```

Returns the frame rectangle used by a window with a given content rectangle and window style.

```
class func minFrameWidth(withTitle: String, styleMask: NSWindow.Style
Mask) -> CGFloat
```

Returns the minimum width a window's frame rectangle must have for it to display a title, with a given window style.

```
func contentRect(forFrameRect: NSRect) -> NSRect
```

Returns the window's content rectangle with a given frame rectangle.

```
func frameRect(forContentRect: NSRect) -> NSRect
```

Returns the window's frame rectangle with a given content rectangle.

Managing Windows

```
var windowController: NSWindowController?
```

The window’s window controller.

Managing Sheets

`var attachedSheet: NSWindow?`

The sheet attached to the window.

`var isSheet: Bool`

A Boolean value that indicates whether the window has ever run as a modal sheet.

`func beginSheet(NSWindow, completionHandler: ((NSApplication.ModalResponse) -> Void)?)`

Starts a document-modal session and presents—or queues for presentation—a sheet.

`func beginCriticalSheet(NSWindow, completionHandler: ((NSApplication.ModalResponse) -> Void)?)`

Starts a document-modal session and presents the specified critical sheet.

`func endSheet(NSWindow)`

Ends a document-modal session and dismisses the specified sheet.

`func endSheet(NSWindow, returnCode: NSApplication.ModalResponse)`

Ends a document-modal session and dismisses the specified sheet.

`var sheetParent: NSWindow?`

The window to which the sheet is attached.

`var sheets: [NSWindow]`

An array of the sheets currently attached to the window.

Sizing Windows

`var frame: NSRect`

The window’s frame rectangle in screen coordinates, including the title bar.

`func setFrameOrigin(NSPoint)`

Positions the bottom-left corner of the window’s frame rectangle at a given point in screen coordinates.

`func setFrameTopLeftPoint(NSPoint)`

Positions the top-left corner of the window's frame rectangle at a given point in screen coordinates.

```
func constrainFrameRect(NSRect, to: NSScreen?) -> NSRect
```

Modifies and returns a frame rectangle so that its top edge lies on a specific screen.

```
func cascadeTopLeft(from: NSPoint) -> NSPoint
```

Positions the window's top-left to a given point.

```
func setFrame(NSRect, display: Bool)
```

Sets the origin and size of the window's frame rectangle according to a given frame rectangle, thereby setting its position and size onscreen.

```
func setFrame(NSRect, display: Bool, animate: Bool)
```

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.

```
func animationResizeTime(NSRect) -> TimeInterval
```

Specifies the duration of a smooth frame-size change.

```
var aspectRatio: NSSize
```

The window's aspect ratio, which constrains the size of its frame rectangle to integral multiples of this ratio when the user resizes it.

```
var minSize: NSSize
```

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

```
var maxSize: NSSize
```

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

```
var isZoomed: Bool
```

A Boolean value that indicates whether the window is in a zoomed state.

```
func performZoom(Any?)
```

This action method simulates the user clicking the zoom box by momentarily highlighting the button and then zooming the window.

```
func zoom(Any?)
```

Toggles the size and location of the window between its standard state (which the application provides 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).

```
var resizeFlags: NSEvent.ModifierFlags
```

The flags field of the event record for the mouse-down event that initiated the resizing session.

`var resizeIncrements: NSSize`

The window’s resizing increments.

`var preservesContentDuringLiveResize: Bool`

A Boolean value that indicates whether the window tries to optimize user-initiated resize operations by preserving the content of views that have not changed.

`var inLiveResize: Bool`

A Boolean value that indicates whether the window is being resized by the user.

Sizing Content

`var contentAspectRatio: NSSize`

The window’s content aspect ratio.

`var contentMinSize: NSSize`

The minimum size of the window’s content view in the window’s base coordinate system.

`func setContentSize(NSSize)`

Sets the size of the window’s content view to a given size, which is expressed in the window’s base coordinate system.

`var contentMaxSize: NSSize`

The maximum size of the window’s content view in the window’s base coordinate system.

`var contentResizeIncrements: NSSize`

The window’s content-view resizing increments.

`var contentLayoutGuide: Any?`

A value used by Auto Layout constraints to automatically bind to the value of `contentLayoutRect`.

`var contentLayoutRect: NSRect`

The area inside the window that is for non-obscured content, in window coordinates.

`var maxFullScreenContentSize: NSSize`

A maximum size that is used to determine if a window can fit when it is in full screen in a tile.

`var minFullScreenContentSize: NSSize`

A minimum size that is used to determine if a window can fit when it is in full screen in a tile.

Managing Window Layers

`func orderOut(Any?)`

Removes the window from the screen list, which hides the window.

`func orderBack(Any?)`

Moves the window to the back of its level in the screen list, without changing either the key window or the main window.

`func orderFront(Any?)`

Moves the window to the front of its level in the screen list, without changing either the key window or the main window.

`func 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.

`func order(NSWindow.OrderingMode, relativeTo: Int)`

Repositions the window’s window device in the window server’s screen list.

`var level: NSWindow.Level`

The window level of the window.

`struct Level`

The standard window levels in macOS.

Managing Window Visibility and Occlusion State

`var isVisible: Bool`

A Boolean value that indicates whether the window is visible onscreen (even when it’s obscured by other windows).

`var occlusionState: NSWindow.OcclusionState`

The occlusion state of the window.

Managing Window Frames in User Defaults

`class func removeFrame(usingName: NSWindow.FrameAutosaveName)`

Removes the frame data stored under a given name from the application’s user defaults.

`func setFrameUsingName(NSWindow.FrameAutosaveName) -> Bool`

Sets the window's frame rectangle by reading the rectangle data stored under a given name from the defaults system.

`func setFrameUsingName(NSWindow.FrameAutosaveName, force: Bool) -> Bool`

Sets the window's frame rectangle by reading the rectangle data stored under a given name from the defaults system. Can operate on non-resizable windows.

`func saveFrame(usingName: NSWindow.FrameAutosaveName)`

Saves the window's frame rectangle in the user defaults system under a given name.

`func setFrameAutosaveName(NSWindow.FrameAutosaveName) -> Bool`

Sets the name AppKit uses to automatically save the window's frame rectangle data in the defaults system.

`var frameAutosaveName: NSWindow.FrameAutosaveName`

The name used to automatically save the window's frame rectangle data in the defaults system.

`typealias FrameAutosaveName`

The type of a window's frame autosave name.

`var frameDescriptor: NSWindow.PersistableFrameDescriptor`

A string representation of the window's frame rectangle.

`func setFrame(from: NSWindow.PersistableFrameDescriptor)`

Sets the window's frame rectangle from a given string representation.

`typealias PersistableFrameDescriptor`

The type of a window's frame descriptor.

Managing Key Status

`var isKeyWindow: Bool`

A Boolean value that indicates whether the window is the key window for the application.

`var canBecomeKey: Bool`

A Boolean value that indicates whether the window can become the key window.

`func makeKey()`

Makes the window the key window.

`func makeKeyAndOrderFront(Any?)`

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.

`func becomeKey()`

Informs the window that it has become the key window.

`func resignKey()`

Resigns the window’s key window status.

Managing Main Status

`var isMainWindow: Bool`

A Boolean value that indicates whether the window is the application’s main window.

`var canBecomeMain: Bool`

A Boolean value that indicates whether the window can become the application’s main window.

`func makeMain()`

Makes the window the main window.

`func becomeMain()`

Informs the window that it has become the main window.

`func resignMain()`

Resigns the window’s main window status.

Managing Toolbars

`var toolbar: NSToolbar?`

The window’s toolbar.

`func toggleToolbarShown(Any?)`

Toggles the visibility of the window’s toolbar.

`func runToolbarCustomizationPalette(Any?)`

Presents the toolbar customization user interface.

Managing Attached Windows

`var childWindows: [NSWindow]?`

An array of the window’s attached child windows.

`func addChildWindow(NSWindow, ordered: NSWindow.OrderingMode)`

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

`func removeChildWindow(NSWindow)`

Detaches a given child window from the window.

`var parent: NSWindow?`

The parent window to which the window is attached as a child.

Managing Default Buttons

`var defaultButtonCell: NSButtonCell?`

The button cell that performs as if clicked when the window receives a Return (or Enter) key event.

`func enableKeyEquivalentForDefaultButtonCell()`

Reenables the default button cell’s key equivalent, so it performs a click when the user presses Return (or Enter).

`func disableKeyEquivalentForDefaultButtonCell()`

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

`func fieldEditor(Bool, for: Any?) -> NSText?`

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

`func endEditing(for: Any?)`

Forces the field editor to give up its first responder status and prepares it for its next assignment.

Managing the Window Menu

`var isExcludedFromWindowsMenu: Bool`

A Boolean value that indicates whether the window is excluded from the application’s Windows menu.

Managing Cursor Rectangles

`var areCursorRectsEnabled: Bool`

A Boolean value that indicates whether the window's cursor rectangles are enabled.

`func enableCursorRects()`

Reenables cursor rectangle management within the window after a `disableCursorRects()` message.

`func disableCursorRects()`

Disables all cursor rectangle management within the window.

`func discardCursorRects()`

Invalidates all cursor rectangles in the window.

`func invalidateCursorRects(for: NSView)`

Marks as invalid the cursor rectangles of a given view object in the window, so they'll be set up again when the window becomes key.

`func resetCursorRects()`

Clears the window's cursor rectangles and the cursor rectangles of the `NSView` objects in its view hierarchy.

Managing Title Bars

`class func standardWindowButton(NSWindow.ButtonType, for: NSWindow.StyleMask) -> NSButton?`

Returns a new instance of a given standard window button, sized appropriately for a given window style.

`func standardWindowButton(NSWindow.ButtonType) -> NSButton?`

Returns the window button of a given window button kind in the window's view hierarchy.

~~`var showsToolbarButton: Bool`~~

A Boolean value that indicates whether the toolbar control button is currently displayed.

Deprecated

`var titlebarAppearsTransparent: Bool`

A Boolean value that indicates whether the title bar draws its background.

`var toolbarStyle: NSWindow.ToolbarStyle`

The style that determines the appearance and location of the toolbar in relation to the title bar.

`enum ToolbarStyle`

Styles that determine the appearance and location of the toolbar in relation to the title bar.

`var titlebarSeparatorStyle: NSTitlebarSeparatorStyle`

The type of separator that the app displays between the title bar and content of a window.

`enum NSTitlebarSeparatorStyle`

Styles that determine the type of separator displayed between the title bar and content of a window.

`var windowTitlebarLayoutDirection: NSUserInterfaceLayoutDirection`

The direction the window’s title bar lays text out, either left to right or right to left.

Managing Title Bar Accessories

`func addTitlebarAccessoryViewController(NSTitlebarAccessoryViewController)`

Adds the specified title bar accessory view controller to the window.

`func insertTitlebarAccessoryViewController(NSTitlebarAccessoryViewController, at: Int)`

Inserts the view controller into the window’s array of title bar accessory view controllers at the specified index.

`func removeTitlebarAccessoryViewController(at: Int)`

Removes the view controller at the specified index from the window’s array of title bar accessory view controllers.

`var titlebarAccessoryViewControllers: [NSTitlebarAccessoryViewController]`

An array of title bar accessory view controllers that are currently added to the window.

Managing Window Tabs

`class var allowsAutomaticWindowTabbing: Bool`

A Boolean value that indicates whether the app can automatically organize windows into tabs.

`class var userTabbingPreference: NSWindow.UserTabbingPreference`

A value that indicates the user’s preference for window tabbing.

`var tab: NSWindowTab`

An object that represents information about a window when it displays as a tab.

`var tabbingIdentifier: NSWindow.TabbingIdentifier`

A value that allows a group of related windows.

`typealias TabbingIdentifier`

A value that allows a group of related windows.

`func addTabbedWindow(NSWindow, ordered: NSWindow.OrderingMode)`

Adds the provided window as a new tab in a tabbed window using the specified ordering instruction.

`var tabbingMode: NSWindow.TabbingMode`

A value that indicates when a window displays tabs.

`var tabbedWindows: [NSWindow]?`

An array of windows that display as tabs.

`func mergeAllWindows(Any?)`

Merges all open windows into a single tabbed window.

`func selectNextTab(Any?)`

Selects the next tab in the tab group in the trailing direction.

`func selectPreviousTab(Any?)`

Selects the previous tab in the tab group in the leading direction.

`func moveTabToNewWindow(Any?)`

Moves the tab to a new containing window.

`func toggleTabBar(Any?)`

Shows or hides the tab bar.

`func toggleTabOverview(Any?)`

Shows or hides the tab overview.

`var tabGroup: NSWindowTabGroup?`

A group of windows that display together as a tab group.

Managing Tooltips

`var allowsToolTipsWhenApplicationIsInactive: Bool`

A Boolean value that indicates whether the window can display tooltips even when the application is in the background.

Handling Events

`var currentEvent: NSEvent?`

The event currently being processed by the application.

`func nextEvent(matching: NSEvent.EventTypeMask) -> NSEvent?`

Returns the next event matching a given mask.

`func nextEvent(matching: NSEvent.EventTypeMask, until: Date?, inMode: RunLoop.Mode, dequeue: Bool) -> NSEvent?`

Forwards the message to the global application object.

`func discardEvents(matching: NSEvent.EventTypeMask, before: NSEvent?)`

Forwards the message to the global application object.

`func postEvent(NSEvent, atStart: Bool)`

Forwards the message to the global application object.

`func sendEvent(NSEvent)`

This action method dispatches mouse and keyboard events the global application object sends to the window.

`func tryToPerform(Selector, with: Any?) -> Bool`

Dispatches action messages with a given argument.

Managing Responders

`var initialFirstResponder: NSView?`

The view that’s made first responder (also called the key view) the first time the window is placed onscreen.

`var firstResponder: NSResponder?`

The window’s first responder.

`func makeFirstResponder(NSResponder?) -> Bool`

Attempts to make a given responder the first responder for the window.

Managing the Key View Loop

`func selectKeyView(preceding: NSView)`

Gives key view status to the view that precedes the given view.

`func selectKeyView(following: NSView)`

Gives key view status to the view that follows the given view.

`func selectPreviousKeyView(Any?)`

Searches for a candidate previous key view and, if it finds one, tries to make it the first responder.

`func selectNextKeyView(Any?)`

Searches for a candidate next key view and, if it finds one, tries to make it the first responder.

`var keyViewSelectionDirection: NSWindow.SelectionDirection`

The direction the window is currently using to change the key view.

`var autorecalculatesKeyViewLoop: Bool`

A Boolean value that indicates whether the window automatically recalculates the key view loop when views are added.

`func recalculateKeyViewLoop()`

Marks the key view loop as “dirty” and in need of recalculation.

Managing Window Sharing

`func transferWindowSharing(to: NSWindow, completionHandler: ((any Error)?) -> Void)`

`var hasActiveWindowSharingSession: Bool`

Handling Mouse Events

`var acceptsMouseMovedEvents: Bool`

A Boolean value that indicates whether the window accepts mouse-moved events.

`var ignoresMouseEvents: Bool`

A Boolean value that indicates whether the window is transparent to mouse events.

```
var mouseLocationOutsideOfEventStream: NSPoint
```

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.

```
class func windowNumber(at: NSPoint, belowWindowWithWindowNumber: Int)
-> Int
```

Returns the number of the frontmost window that would be hit by a mouse-down at the specified screen location.

```
func trackEvents(matching: NSEvent.EventTypeMask, timeout: TimeInterval
, mode: RunLoop.Mode, handler: (NSEvent?, UnsafeMutablePointer<ObjCBool
>) -> Void)
```

Tracks events that match the specified mask using the specified tracking handler until the tracking handler explicitly terminates tracking.

```
func performDrag(with: NSEvent)
```

Starts a window drag based on the specified mouse-down event.

```
class let foreverDuration: TimeInterval
```

The longest time duration possible.

Handling Window Restoration

```
var isRestorable: Bool
```

A Boolean value indicating whether the window configuration is preserved between application launches.

```
var restorationClass: (any NSWindowRestoration.Type)?
```

The restoration class associated with the window.

```
func disableSnapshotRestoration()
```

Disables snapshot restoration.

```
func enableSnapshotRestoration()
```

Enables snapshot restoration.

Drawing Windows

```
func display()
```

Passes a display message down the window's view hierarchy, thus redrawing all views within the window.

`func displayIfNeeded()`

Passes a display message down the window's view hierarchy, thus redrawing all views that need displaying.

`var viewsNeedDisplay: Bool`

A Boolean value that indicates whether any of the window's views need to be displayed.

`var allowsConcurrentViewDrawing: Bool`

A Boolean value that indicates whether the window allows multithreaded view drawing.

Window Animation

`var animationBehavior: NSWindow.AnimationBehavior`

The window's automatic animation behavior.

Updating Windows

~~`func disableScreenUpdatesUntilFlush()`~~

Disables the window's screen updates until the window is flushed.

Deprecated

`func update()`

Updates the window.

Dragging Items

~~`func drag(NSImage, at: NSPoint, offset: NSSize, event: NSEvent, pasteboard: NSPasteboard, source: Any, slideBack: Bool)`~~

Begins a dragging session.

Deprecated

`func registerForDraggedTypes([NSPasteboard.PasteboardType])`

Registers a set of pasteboard types that the window accepts as the destination of an image-dragging session.

`func unregisterDraggedTypes()`

Unregisters the window as a possible destination for dragging operations.

Accessing Edited Status

`var isDocumentEdited: Bool`

A Boolean value that indicates whether the window’s document has been edited.

Converting Coordinates

`var backingScaleFactor: CGFloat`

The backing scale factor.

`func backingAlignedRect(NSRect, options: AlignmentOptions) -> NSRect`

Returns a backing store pixel-aligned rectangle in window coordinates.

`func convertFromBacking(NSRect) -> NSRect`

Converts a rectangle from its pixel-aligned backing store coordinate system to the window’s coordinate system.

`func convertFromScreen(NSRect) -> NSRect`

Converts a rectangle from the screen coordinate system to the window’s coordinate system.

`func convertPointFromBacking(NSPoint) -> NSPoint`

Converts a point from its pixel-aligned backing store coordinate system to the window’s coordinate system.

`func convertPoint(fromScreen: NSPoint) -> NSPoint`

Converts a point from the screen coordinate system to the window’s coordinate system.

`func convertToBacking(NSRect) -> NSRect`

Converts a rectangle from the window’s coordinate system to its pixel-aligned backing store coordinate system.

`func convertToScreen(NSRect) -> NSRect`

Converts a rectangle to the screen coordinate system from the window’s coordinate system.

`func convertPointToBacking(NSPoint) -> NSPoint`

Converts a point from the window’s coordinate system to its pixel-aligned backing store coordinate system.

`func convertPoint(toScreen: NSPoint) -> NSPoint`

Converts a point to the screen coordinate system from the window’s coordinate system.

Managing Titles

`var title: String`

The string that appears in the title bar of the window or the path to the represented file.

`var subtitle: String`

A secondary line of text that appears in the title bar of the window.

`var titleVisibility: NSWindow.TitleVisibility`

A value that indicates the visibility of the window’s title and title bar buttons.

`func setTitleWithRepresentedFilename(String)`

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 file.

`var representedFilename: String`

The path to the file of the window’s represented file.

`var representedURL: URL?`

The URL of the file the window represents.

Accessing Screen Information

`var screen: NSScreen?`

The screen the window is on.

`var deepestScreen: NSScreen?`

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

`var displaysWhenScreenProfileChanges: Bool`

A Boolean value that indicates whether the window context should be updated when the screen profile changes or when the window moves to a different screen.

Moving Windows

`var isMovableByWindowBackground: Bool`

A Boolean value that indicates whether the window is movable by clicking and dragging anywhere in its background.

`var isMovable: Bool`

A Boolean value that indicates whether the window can be dragged by clicking in its title bar or background.

`func center()`

Sets the window's location to the center of the screen.

Closing Windows

`func performClose(Any?)`

Simulates the user clicking the close button by momentarily highlighting the button and then closing the window.

`func close()`

Removes the window from the screen.

`var isReleasedWhenClosed: Bool`

A Boolean value that indicates whether the window is released when it receives the `close` message.

Minimizing Windows

`var isMiniaturized: Bool`

A Boolean value that indicates whether the window is minimized.

`func performMiniaturize(Any?)`

Simulates the user clicking the minimize button by momentarily highlighting the button, then minimizing the window.

`func miniaturize(Any?)`

Removes the window from the screen list and displays the minimized window in the Dock.

`func deminiaturize(Any?)`

De-minimizes the window.

`var miniwindowImage: NSImage?`

The custom miniaturized window image of the window.

`var miniwindowTitle: String!`

The title displayed in the window's minimized window.

Getting the Dock Tile

`var dockTile: NSDockTile`

The application’s Dock tile.

Printing Windows

`func printWindow(Any?)`

Runs the Print panel, and if the user chooses an option other than canceling, prints the window (its frame view and all subviews).

`func dataWithEPS(inside: NSRect) -> Data`

Returns EPS data that draws the region of the window within a given rectangle.

`func dataWithPDF(inside: NSRect) -> Data`

Returns PDF data that draws the region of the window within a given rectangle.

Providing Services

`func validRequestor(forSendType: NSPasteboard.PasteboardType?, returnType: NSPasteboard.PasteboardType?) -> Any?`

Searches for an object that responds to a Services request.

Triggering Constraint-Based Layout

`func updateConstraintsIfNeeded()`

Updates the constraints based on changes to views in the window since the last layout.

`func layoutIfNeeded()`

Updates the layout of views in the window based on the current views and constraints.

Debugging Constraint-Based Layout

See for more details on debugging constraint-based layout.

`func visualizeConstraints([NSLayoutConstraint]?)`

Displays a visual representation of the supplied constraints in the window.

Constraint-Based Layouts

```
func anchorAttribute(for: NSLayoutConstraint.Orientation) -> NSLayout
Constraint.Attribute
```

Returns the part of the window that stays stationary during constraint-based layout.

```
func setAnchorAttribute(NSLayoutConstraint.Attribute, for: NSLayout
Constraint.Orientation)
```

Sets the part of the window that stays stationary during constraint-based layout.

Working with Window Depths

```
var bitsPerPixel: Int
```

Returns the bits per pixel for the specified window depth.

```
var bitsPerSample: Int
```

Returns the bits per sample for the specified window depth.

```
var colorSpaceName: NSColorSpaceName?
```

Returns the name of the color space corresponding to the passed window depth.

```
var numberOfColorComponents: Int
```

Returns the number of color components in the specified color space.

```
var isPlanar: Bool
```

Returns whether the specified window depth is planar.

```
func canRepresent(NSDisplayGamut) -> Bool
```

A Boolean value that indicates if the window and its screen use a color space that can represent the specified display gamut.

Getting Information About Scripting Attributes

```
var hasCloseBox: Bool
```

A Boolean value that indicates if the window has a close box.

```
var hasTitleBar: Bool
```

A Boolean value that indicates if the window has a title bar.

```
var isModalPanel: Bool
```

A Boolean value that indicates whether the window is a modal panel.

```
var isFloatingPanel: Bool
```

A Boolean value that indicates whether the window is a floating panel.

`var isZoomable: Bool`

A Boolean value that indicates whether the window allows zooming.

`var isResizable: Bool`

A Boolean value that indicates if the user can resize the window.

`var isMiniaturizable: Bool`

A Boolean value that indicates whether the window can minimize.

`var orderedIndex: Int`

The zero-based position of the window, based on its order from front to back among all visible application windows.

Setting Scripting Attributes

`func setIsMiniaturized(Bool)`

Sets the window's miniaturized state to the value you specify.

`func setIsVisible(Bool)`

Sets the window's visible state to the value you specify.

`func setIsZoomed(Bool)`

Sets the window's zoomed state to the value you specify.

Handling Script Commands

`func handleClose(NSCloseCommand) -> Any?`

Handles the AppleScript command to close the window (and its associated document, if any).

`func handlePrint(NSScriptCommand) -> Any?`

Handles the AppleScript command to print the contents of the window (or its associated document, if any).

`func handleSave(NSScriptCommand) -> Any?`

Handles the AppleScript command to save the window (and its associated document, if any).

Constants

`enum SelectionDirection`

Constants that specify the direction a window is currently using to change the key view.

`enum ButtonType`

Constants that provide a way to access standard title bar buttons.

NSRunLoop—Ordering Modes for NSWindow

Constants that specify the priority for runloop messages.

`enum Depth`

A type that represents the depth, or amount of memory, for a single pixel in a window or screen.

`enum BackingStoreType`

Constants that specify how the window device buffers the drawing done in a window.

`enum OrderingMode`

Constants that let you specify how a window is ordered relative to another window.

`enum SharingType`

Constants that represent the access levels other processes can have to a window’s content.

`struct NumberListOptions`

Options to use when retrieving window numbers from the system.

`enum AnimationBehavior`

Constants that control the automatic window animation behavior windows use when ordering to the front or out of view.

`struct CollectionBehavior`

Window collection behaviors related to Mission Control, Spaces, and Stage Manager.

`struct OcclusionState`

Specifies whether the window is occluded.

`enum TitleVisibility`

Specifies the appearance of the window’s title bar area.

`enum UserTabbingPreference`

A value that indicates the user’s preference for window tabbing.

`enum TabbingMode`

The preferred tabbing behavior of a window.

☰ Application Kit Version for Deferred Window Display Support

The version of the AppKit.framework containing a specific bug fix or capability.

☰ Application Kit Version for Custom Sheet Position

The version of the AppKit.framework containing a specific bug fix or capability.

☰ NSWindowDidChangeBackingPropertiesNotification User Info Properties

Constants that represent values in the user info dictionary of the `didChangeBackingPropertiesNotification` notification.

Notifications

`class let didBecomeKeyNotification: NSNotification.Name`

A notification that the window object became the key window.

`class let didBecomeMainNotification: NSNotification.Name`

A notification that the window object became the main window.

`class let didChangeScreenNotification: NSNotification.Name`

A notification that a portion of the window object's frame moved onto or off of a screen.

`class let didChangeScreenProfileNotification: NSNotification.Name`

A notification that the screen containing the window changed.

`class let didDeminiaturizeNotification: NSNotification.Name`

A notification that the window is no longer minimized.

`class let didEndSheetNotification: NSNotification.Name`

A notification that the window object closed an attached sheet.

`class let didEndLiveResizeNotification: NSNotification.Name`

A notification that the user resized the window object.

`class let didExposeNotification: NSNotification.Name`

A notification that a window exposed a portion of its nonretained content.

`class let didMiniaturizeNotification: NSNotification.Name`

A notification that the window object minimized.

`class let didMoveNotification: NSNotification.Name`

A notification that the window object moved.

`class let didResignKeyNotification: NSNotification.Name`

A notification that the window object resigned its status as key window.

`class let didResignMainNotification: NSNotification.Name`

A notification that the window object resigned its status as main window.

`class let didResizeNotification: NSNotification.Name`

A notification that the window object size changed.

`class let didUpdateNotification: NSNotification.Name`

A notification that the window object received an update message.

`class let willBeginSheetNotification: NSNotification.Name`

A notification that the window object is about to open a sheet.

`class let willCloseNotification: NSNotification.Name`

A notification that the window object is about to close.

`class let willMiniaturizeNotification: NSNotification.Name`

A notification that the window object is about to minimize.

`class let willMoveNotification: NSNotification.Name`

A notification that the window object is about to move.

`class let willStartLiveResizeNotification: NSNotification.Name`

A notification that the user is about to resize the window.

`class let willEnterFullScreenNotification: NSNotification.Name`

A notification that the window will enter full-screen mode.

`class let didEnterFullScreenNotification: NSNotification.Name`

A notification that the window entered full-screen mode.

`class let willExitFullScreenNotification: NSNotification.Name`

A notification that the window object will exit full-screen mode.

`class let didExitFullScreenNotification: NSNotification.Name`

A notification that the window object exited full-screen mode.

`class let willEnterVersionBrowserNotification: NSNotification.Name`

A notification that the window object will enter version browser mode.

`class let didEnterVersionBrowserNotification: NSNotification.Name`

A notification that the window object entered version browser mode.

`class let willExitVersionBrowserNotification: NSNotification.Name`

A notification that the window object will exit version browser mode.

`class let didExitVersionBrowserNotification: NSNotification.Name`

A notification that the window object exited version browser mode.

`class let didChangeBackingPropertiesNotification: NSNotification.Name`

A notification that the window object backing properties changed.

`class let didChangeOcclusionStateNotification: NSNotification.Name`

A notification that the window object's occlusion state changed.

Deprecated

Deprecated Symbols

Review unsupported symbols and their replacements.

Classes

`class HostingSheetRepresentation`

A class representing a SwiftUI view hosted in an AppKit sheet.

Instance Properties

`var cascadingReferenceFrame: NSRect`

Instance Methods

```
func beginDraggingSession(items: [NSDraggingItem], event: NSEvent,
source: any NSDraggingSource) -> NSDraggingSession
```

```
func beginSheet<V>(content: () -> V, completionHandler: (() -> Void)?)
-> NSWindow.HostingSheetRepresentation<V>
```

Presents a SwiftUI View as a sheet on the receiving NSWindow.

`func endSheet<V>(NSWindow.HostingSheetRepresentation<V>)`

Ends a SwiftUI hosted sheet presentation.

```
func requestSharingOfWindow(NSWindow, completionHandler: ((any Error)?)
-> Void)
```

```
func requestSharingOfWindow(usingPreview: NSImage, title: String,
completionHandler: ((any Error)?) -> Void)
```

Relationships

Inherits From

`NSResponder`

Inherited By

`NSPanel`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSAccessibilityElementProtocol`
`NSAccessibilityProtocol`
`NSAnimatablePropertyContainer`
`NSAppearanceCustomization`
`NSCoding`
`NSMenuItemValidation`
`NSObjectProtocol`
`NSStandardKeyBindingResponding`
`NSTouchBarProvider`
`NSUserActivityRestoring`
`NSUserInterfaceItemIdentification`
`NSUserInterfaceValidations`
`Sendable`
`SendableMetatype`

See Also

Windows

`class NSPanel`

A special kind of window that typically performs a function that is auxiliary to the main window.

`protocol NSWindowDelegate`

A set of optional methods that a window’s delegate can implement to respond to events, such as window resizing, moving, exposing, and minimizing.

`class NSWindowTab`

A tab associated with a window that is part of a tabbing group.

`class NSWindowTabGroup`

A group of windows that display together as a single tabbed window.