



# CocoaUI



# Table of Contents

## Introduction & General Overview

Interface elements .....	7
Events .....	7
Requirements .....	7
Demos .....	7
Scripts .....	7

## Working with Nibs

## Reference

AffineTransform .....	10
CocoaAlert .....	11
Appearance .....	13
AppleEventDescriptor .....	14
AppleScript .....	15
Application .....	16
CocoaInit .....	19
Array .....	20
Asset .....	22
AssetTrack .....	23
AttributedString .....	24
AVPlayer .....	29
AVPlayerItem .....	30
AVPlayerLayer .....	31
AVPlayerView .....	32
BezierPath .....	33
BitmapImageRep .....	35
Cocoa Box .....	36
Bundle .....	38
Cocoa Button .....	39
ByteCountFormatter .....	42
CAAnimation .....	43
CABasicAnimation .....	44
CAGradientLayer .....	45
CAKeyframeAnimation .....	46
CALayer .....	47
Calendar .....	49
CAMediaTiming .....	51
CAMediaTimingFunction .....	52
CAPropertyAnimation .....	53
CAReplicatorLayer .....	54
CAScrollLayer .....	55
CAShapeLayer .....	56
CASpringAnimation .....	57
CATextLayer .....	58
CATiledLayer .....	59
CATransform3D .....	60
CATransformLayer .....	61
CATransition .....	62
CAValueFunction .....	63
CharacterSet .....	64
CIContext .....	65
CIDetector .....	66

CIFeature .....	67
CIFilter.....	68
CImage.....	69
CImageRep .....	70
CIQRCodeFeature .....	71
ClickGestureRecognizer .....	72
ClipView .....	73
CMTIME .....	74
Coder .....	75
Color .....	76
ColorList.....	80
ColorPanel .....	81
ColorList.....	82
ColorWell.....	83
ComboBox.....	84
ComparisonPredicate .....	86
Compiler .....	87
CompoundPredicate.....	88
Control .....	89
CountedSet.....	90
Cursor .....	91
Data .....	92
DataDetector.....	93
Date .....	94
DateComponents.....	95
DateComponentsFormatter .....	96
DateFormatter.....	97
DateInterval.....	99
DateIntervalFormatter .....	100
DatePicker .....	101
DecimalNumber .....	103
DecimalNumberBehaviors .....	104
DecimalNumberHandler .....	105
Dialog.....	106
Dictionary .....	107
DirectoryEnumerator.....	109
DistributedNotificationCenter .....	110
DockTile.....	111
DraggingInfo .....	112
EditMenu.....	113
Enumerator .....	114
Error .....	115
Event.....	116
Expression .....	117
FileHandle .....	118
FileManager .....	119
FileMenu .....	121
FileWrapper.....	122
FindMenu .....	123
Font.....	124
FontDescriptor .....	125
FontManager.....	126
FontMenu.....	128
FormatMenu .....	129
Geometry .....	130
GestureRecognizer .....	131
GlyphGenerator .....	132
GlyphStorage .....	133

Gradient .....	134
Graphics .....	135
GraphicsContext.....	136
HelpManager .....	137
HelpMenu .....	138
Image .....	139
ImageRep.....	141
ImageView .....	142
IndexPath.....	144
IndexSet.....	145
JSONSerialization .....	146
KeyedArchiver.....	147
KeyedUnarchiver .....	148
LayoutManager.....	149
LevelIndicator .....	151
LinguisticTagger .....	154
Locale .....	155
MagnificationGestureRecognizer.....	157
MKMapView.....	158
MenuEvent.....	160
Cocoa Menu .....	161
MenuItem .....	163
NibMenu .....	165
NibPopover .....	166
NibToolbar .....	167
NibView.....	168
NibWindow .....	169
Notification.....	170
NotificationCenter .....	171
NotificationQueue .....	172
NSDecimal .....	173
NSHFSFileTypes .....	174
Number .....	175
NumberFormatter .....	176
Object .....	179
ObjCRuntime .....	180
OpenPanel .....	181
Operation .....	183
OperationQueue .....	184
OrderedSet .....	185
Orthography.....	187
OutlineView .....	188
PageLayout.....	190
Panel .....	191
PanGestureRecognizer .....	192
ParagraphStyle .....	193
Pasteboard .....	195
PathUtilities.....	196
PDFDocument .....	197
PDFInfo.....	199
PDFPanel .....	200
PDFThumbnailView .....	201
PDFView .....	202
Pipe.....	205
PointerArray .....	206
PointerFunctions.....	207
Popover .....	208
PopUpButton .....	210

Predicate.....	212
PressGestureRecognizer .....	213
PressureConfiguration .....	214
Printer .....	215
PrintInfo .....	216
PrintOperation.....	217
PrintPanel.....	218
ProcessInfo .....	219
ProgressIndicator.....	220
PropertyListSerialization .....	223
Range.....	224
RegularExpression .....	225
Responder .....	226
RotationGestureRecognizer.....	227
RulerMarker .....	228
RulerView .....	229
RunningApplication.....	230
SavePanel .....	231
Scanner.....	233
Screen.....	234
Scroller.....	235
ScrollView .....	236
SearchField .....	238
SecureTextField .....	239
SegmentedControl.....	240
Set.....	242
Shadow.....	243
Slider.....	244
SortDescriptor.....	246
Sound .....	247
SplitView .....	248
StatusBar .....	250
StatusBarButton .....	251
StatusItem.....	252
Stepper .....	253
String .....	254
StringDrawingContext .....	257
Subclass .....	258
TableRowView.....	259
TableView .....	260
TabView.....	263
Task.....	265
Text .....	266
TextAttachment.....	268
TextBlock .....	269
TextCheckingResult .....	270
TextContainer .....	272
TextField.....	273
TextFinder .....	275
TextLabel.....	276
TextList.....	277
TextMenu .....	278
TextStorage.....	279
TextTab.....	280
TextTable .....	281
TextTableBlock.....	282
TextView.....	283
Thread.....	286

Timer .....	287
TimeZone .....	288
TokenField .....	289
Toolbar .....	291
ToolbarItem .....	292
Touch .....	294
UndoManager .....	295
URL .....	296
URLAsset .....	297
URLSession .....	298
URLSessionConfiguration .....	300
URLSessionDownloadTask .....	301
URLSessionStreamTask .....	302
URLSessionTask .....	303
UserDefaults .....	304
UserDefaultsController .....	305
UserNotification .....	306
UserNotificationAction .....	307
UserNotificationCenter .....	308
va_list .....	309
Value .....	310
View .....	311
ViewController .....	315
ViewMenu .....	316
VisualEffectView .....	317
WKWebView .....	318
Cocoa Window .....	320
WindowTab .....	326
WindowTabGroup .....	327
WindowMenu .....	328
Workspace .....	329



# Introduction & General Overview

---

CocoaUI supplements FB's current syntax and offers a potential transition path for Carbon-based code. CocoaUI has its own set of reserved words to create interface elements.

## Interface elements

### Windows

Windows are created with the **cocoa window** statement and are positioned relative to the bottom-left of the screen. Although Cocoa windows can coexist in the same app as Carbon windows, they can only contain Cocoa widgets.

### Widgets

Widgets, for the most part, have their own keyword to construct in code. A Cocoa widget is positioned relative to the bottom-left of its parent view (usually the window's content view). The majority of Cocoa widgets work fine in Carbon apps but they cannot be placed in Carbon windows.

### Menus and Menu Items

Menus and menu items are created using the **cocoa menu** statement. Menu item indexes are zero-based. Cocoa menus must have [CocoaInit](#).

## Events

**Dialog** events are handled automatically and intercepted in the traditional 'on dialog fn xxxxx' function. With the exception of \_btnClick, CocoaUI elements have their own dialog event constants.

**Menu** events are handled in the usual 'on menu fn xxxx' function. Note: Cocoa menu item indexes are zero-based and Cocoa menus require [CocoaInit](#) statement.

**Application** events are handled in CocoaUI's 'on appevent fn xxxx' function and require [CocoaInit](#).

The **Cocoa event loop** is initiated by inserting the [CocoaInit](#) statement at the top of the app. A call to FB's HandleEvents will do the rest.

Note: An app cannot have *both* Cocoa and Carbon event loops (e.g. RunApplicationEventLoop).

## Requirements

- FB 5.7.104 or later
- include "Tlbx CocoaUI.incl"
- Build Settings 'Base SDK' 10.7 or higher

## Demos

Demos showing how to create and manipulate the majority of CocoaUI elements can be found in: FBExamples/CocoaUI Demos.

## Scripts

A set of FB Scripts have been created to assist developers. The CocoaUI scripts folder should be placed in ~/Library/Application Support/Scripts. The scripts will appear in FB's scripts palette on the next launch of the FutureBasic editor. The latest CocoaUI scripts can be installed by choosing 'InstallCocoaUIScripts.app' from FB's Help menu.



# Working with Nibs

---

## About nibs

With nib files, you create and manipulate your user interfaces graphically, using Xcode, instead of programmatically. Because you can see the results of your changes instantly, you can experiment with different layouts and configurations very quickly. You can also change many aspects of your user interface later without rewriting any code.

## Subclassing nib widgets

To subclass a widget contained in a nib, change the widget's class name in the identity inspector. For example, to subclass a nib textfield, set its class name to "TextFieldSC". Note: class names are case-sensitive and are made up of the widget name with "SC" suffix.

## Tutorials

[Working with nibs](#)





# Reference

---

## **Keywords/Functions**

The following pages describe the various CocoaUI keywords/functions and their parameters. Note that most keyword parameters are optional and, if omitted, will default to appropriate values.

Most functions identify a UI element by its tag value. However, an element's reference can also be used in the function's tag parameter.

## **Autoreleased Objects**

ALL CocoaUI functions return autoreleased objects so must not be CFRelease(d). Even functions with names containing the words 'init', 'create' or 'copy' return autoreleased objects.



---

## AffineTransform

### Functions

Create

```
AffineTransformInit = AffineTransformRef// autoreleased  
AffineTransformWithTransform( AffineTransformRef ref ) = AffineTransformRef
```

Accumulating

```
AffineTransformRotateByDegrees( AffineTransformRef tx, CGFloat angle )  
AffineTransformRotateByRadians( AffineTransformRef tx, CGFloat angle )  
AffineTransformScale( AffineTransformRef tx, CGFloat scale )  
AffineTransformScaleXY( AffineTransformRef tx, CGFloat xScale, CGFloat yScale )  
AffineTransformTranslate( AffineTransformRef tx, CGFloat x, CGFloat y )  
AffineTransformAppend( AffineTransformRef tx1, AffineTransformRef tx2 )  
AffineTransformPrepend( AffineTransformRef tx1, AffineTransformRef tx2 )  
AffineTransformInvert( AffineTransformRef tx )
```

Data and objects

```
AffineTransformPoint( AffineTransformRef tx, CGPoint pt ) = CGPoint  
AffineTransformSize( AffineTransformRef tx, CGSize size ) = CGSize  
AffineTransformBezierPath( AffineTransformRef tx, BezierPathRef path ) = BezierPathRef
```

Matrix

```
AffineTransformStruct( AffineTransformRef tx ) = NSAffineTransformStruct
```

Setting and building

```
AffineTransformSet( AffineTransformRef tx )  
AffineTransformConcat( AffineTransformRef tx )
```

### Apple documentation

[NSAffineTransform](#)



CocoaAlert

function/statement

Syntax

```
[ 1 ]
(NSInteger) response = cocoaalert tag, style, msg, info, buttonTitles, sheetFlag

[ 2 ]
cocoaalert tag, style, msg, info, buttonTitles, sheetFlag
```

Description

- Use this statement to:
- Create a new cocoa alert dialog;
  - Show or hide an existing cocoa alert;
  - Alter an existing alert's characteristics.

Note: non-sheet alerts require the Cocoa runtime ([CocoaInit](#)).

Parameters

<i>tag</i>	Unique number to identify the alert dialog.  A negative tag value allows the the alert to be built invisibly so that additional features may be added.  Once an alert has been shown, issuing a further cocoaalert statement with negative tag closes a sheet alert but has no affect on a standard alert.
<i>style</i>	The alert style. <a href="#">NSWarningAlertStyle</a> (default) <a href="#">NSInformationalAlertStyle</a> <a href="#">NSCriticalAlertStyle</a>
<i>msg</i>	The alert message text.
<i>info</i>	The alert informative text.
<i>buttonTitles</i>	A semicolon-delimited string or array of button titles.
<i>sheetFlag</i>	If this param is <code>_true</code> , the alert will be a sheet attached to the current output window. Note: An alert sheet does not return a response value from the cocoaalert statement, and instead posts an <code>_alertDidEnd</code> event to the user on dialog function.

Return value

<code>NSAlertFirstButtonReturn</code>	The user clicked the first (rightmost) vutton on the dialog or sheet.
<code>NSAlertSecondButtonReturn</code>	The user clicked the second button from the right edge of the dialog or sheet.
<code>NSAlertThirdButtonReturn</code>	The user clicked the third button from the right edge of the dialog or sheet.

Dialog Events

Event	Description
<code>_alertDidEnd</code>	This event is triggered when the alert dialog is dismissed.
<code>_alertShowHelp</code>	If the alert is set to show the Help button, this event is triggered when the Help button is clicked. If the alert has a help anchor, the application's help manager displays help using the help anchor. Call <b>DialogEventSetBool</b> ( <code>_true</code> ) if help has been displayed directly.

## Functions

```
AlertWithTag( NSInteger tag ) = AlertRef  
AlertExists( NSInteger tag ) = Boolean
```

### Configure

```
AlertLayout( NSInteger tag )  
AlertStyle( NSInteger tag ) = NSAlertStyle  
AlertSetStyle( NSInteger tag, NSAlertStyle style )  
AlertSetAccessoryView( NSInteger alertTag, viewTag )  
AlertSetShowsHelp( NSInteger tag, Boolean flag )  
AlertHelpAnchor( NSInteger tag ) = CFStringRef  
AlertSetHelpAnchor( NSInteger tag, CFStringRef anchorName )
```

### Displaying

```
AlertSetShowsSuppressionButton( NSInteger tag, Boolean flag )
```

### Text

```
AlertInformativeText( NSInteger tag ) = CFStringRef  
AlertSetInformativeText( NSInteger tag, CFStringRef string )  
AlertMessageText( NSInteger tag ) = CFStringRef  
AlertSetMessageText( NSInteger tag, CFStringRef string )
```

### Icon

```
AlertSetIcon( NSInteger tag, ImageRef image )  
AlertSetIconNamed( NSInteger tag, CFStringRef imageName )
```

### Buttons

```
AlertAddButtonWithTitle( NSInteger tag, CFStringRef title )
```

### Window

```
AlertWindow( NSInteger tag ) = CocoaWindowRef
```

### Convenience

```
AlertSuppressionButtonState( NSInteger tag ) = Boolean  
AlertSuppressionButtonSetState( NSInteger tag, NSControlStateValue value )  
AlertSuppressionButtonSetTitle( NSInteger tag, CFStringRef title )  
AlertClose( NSInteger tag )// macOS 10.9+  
AlertRemove( NSInteger tag )  
AlertButtonSetKeyEquivalent( NSInteger alertTag, NSUInteger btnIndex, CFStringRef key )  
AlertButtonSetKeyEquivalentModifierMask( NSInteger alertTag, NSUInteger btnIndex, NSUInteger mask )
```

## Apple documentation

[NSAlert](#)



---

## Appearance

Requires:  
macOS 10.10+

### Functions

Name

```
AppearanceName( AppearanceRef ref ) = CFStringRef
```

Appearance

```
AppearanceNamed( CFStringRef name ) = AppearanceRef
```

Current appearance

```
AppearanceCurrent = AppearanceRef  
AppearanceSetCurrent( AppearanceRef ref )
```

Vibrancy

```
AppearanceAllowsVibrancy( AppearanceRef ref ) = Boolean
```

### Apple documentation

[NSAppearance](#)



---

## AppleEventDescriptor

### Functions

Info

```
AppleEventDescriptorAEDesc( AppleEventDescriptorRef aed ) = ptr// *AEDesc
AppleEventDescriptorBooleanValue( AppleEventDescriptorRef aed ) = Boolean
AppleEventDescriptorCoerceToType( AppleEventDescriptorRef aed, DescType type ) = AppleEventDescriptorRef
AppleEventDescriptorData( AppleEventDescriptorRef aed ) = CFDataRef
AppleEventDescriptorType( AppleEventDescriptorRef aed ) = DescType
AppleEventDescriptorEnumCodeValue( AppleEventDescriptorRef aed ) = OSType
AppleEventDescriptorInt32Value( AppleEventDescriptorRef aed ) = SInt32
AppleEventDescriptorNumberOfItems( AppleEventDescriptorRef aed ) = NSInteger
AppleEventDescriptorStringValue( AppleEventDescriptorRef aed ) = CFStringRef
AppleEventDescriptorTypeCodeValue( AppleEventDescriptorRef aed ) = OSType
```

### Apple documentation

[NSAppleEventDescriptor](#)



---

## AppleScript

### Functions

Init

```
AppleScriptWithContentsOfURL( CFURLRef url, CFDictionaryRef *errInfo ) = AppleScriptRef  
AppleScriptWithSource( CFStringRef source ) = AppleScriptRef
```

Info

```
AppleScriptIsCompiled( AppleScriptRef script ) = Boolean  
AppleScriptSource( AppleScriptRef script ) = CFStringRef  
AppleScriptRichTextSource( AppleScriptRef script ) = CFAttributedStringRef
```

Compile and execute

```
AppleScriptCompile( AppleScriptRef script, CFDictionaryRef *errInfo ) = Boolean  
AppleScriptExecute( AppleScriptRef script, CFDictionaryRef *errInfo ) = AppleEventDescriptorRef  
AppleScriptExecuteAppleEvent( AppleScriptRef script, AppleEventDescriptorRef aeDesc, CFDictionaryRef *errorInfo ) =  
AppleEventDescriptorRef
```

### Apple documentation

[NSAppleScript](#)



---

## Application

### App Events

```
_appWillFinishLaunching
_appDidFinishLaunching
_appShouldTerminate
_appShouldTerminateAfterLastWindowClosed
_appWillTerminate
_appWillBecomeActive
_appDidBecomeActive
_appWillResignActive
_appDidResignActive
_appWillHide
_appDidHide
_appWillUnhide
_appDidUnhide
_appWillUpdate
_appDidUpdate
_appShouldHandleReopen
_appDockMenu
_appWillPresentError
_appDidChangeScreenParameters
_appOpenFileWithoutUI
_appOpenTempFile
_appOpenFiles
_appOpenUntitledFile
_appShouldOpenUntitledFile
_appPrintFile
_appPrintFiles
_appDidRegisterForRemoteNotificationsWithDeviceToken
_appDidFailToRegisterForRemoteNotifications
_appDidReceiveRemoteNotification
_appDidDecodeRestorableState
_appWillEncodeRestorableState
_appWillContinueUserActivityWithType
_appContinueUserActivity
_appDidFailToContinueUserActivityWithType
_appDidUpdateUserActivity
_appDidChangeOcclusionState
_appOpenURLs
```

### App Event Functions

```
AppEventSetBool( Boolean value )
AppEventSetLong( long value )
AppEventString = CFStringRef
AppEventArray = CFArrayRef
AppEventDictionary = CFDictionaryRef
AppEventURL = CFURLRef
AppEventSetMenu( MenuRef m )
```

### Functions

```
Shared app
AppSharedApplication = ApplicationRef
```

#### Configure

```
AppIconImage = ImageRef
AppSetIconImage( ptr image )
```

#### Terminate

```
AppTerminate
AppReplyToApplicationShouldTerminate( Boolean flag )
```

#### Handling events

```
AppCurrentEvent = CocoaEventRef
```

#### Event loop

```
AppIsRunning = Boolean
AppRun
AppStop
AppSendEvent( CocoaEventRef evnt )
AppPostEvent( CocoaEventRef evnt, Boolean atStart )
```



#### Remote notifications

`AppRegisterForRemoteNotifications` // macOS 10.14+

#### Modal windows

`AppRunModal( NSInteger wndTag ) = NSInteger`

`AppStopModal`

`AppStopModalWithCode( NSInteger returnCode )`

`AppAbortModal`

`AppBeginModalSession( NSInteger wndTag ) = NSModalSession`

`AppRunModalSession( NSModalSession session ) = NSModalResponse` // macOS 10.9+

`AppModalWindow = NSInteger`

`AppEndModalSession( NSModalSession session )`

#### Panels

`AppOrderFrontColorPanel`

`AppOrderFrontStandardAboutPanel`

`AppOrderFrontStandardAboutPanelWithOptions( NSDictionaryRef options )`

`AppOrderFrontCharacterPalette`

`AppRunPageLayout`

#### UI layout direction

`AppUserInterfaceLayoutDirection = NSUserInterfaceLayoutDirection`

#### Managing windows

`AppKeyWindow = NSInteger`

`AppMainWindow = NSInteger`

`AppWindowWithWindowNumber( NSInteger number ) = CocoaWindowRef`

`AppWindows = CFArrayRef`

`AppEnumerateWindows( NSWindowListOptions options, ptr callback, ptr userData )` // macOS 10.12+

#### Minimizing windows

`AppMiniaturizeAll`

#### Hiding windows

`AppIsHidden = Boolean`

`AppHide`

`AppUnhide`

`AppUnhideWithoutActivation`

#### Windows layers

`AppArrangeInFront`

#### Main menu

`AppMainMenu = CocoaMenuRef`

#### Window menu

`AppWindowsMenu = CocoaMenuRef`

#### Appearance

`AppAppearance = AppearanceRef` // macOS 10.14+

`AppEffectiveAppearance = AppearanceRef` // macOS 10.14+

#### Dock tile

`AppDockTile = DockTileRef`

#### Activation status

`AppIsActive = Boolean`

`AppActivateIgnoringOtherApps( Boolean flag )`

`AppDeactivate`

#### Hiding apps

`AppHideOtherApplications`

`AppUnhideAllApplications`

#### Displaying help

`AppShowHelp`

`AppHelpMenu = CocoaMenuRef`

#### User attention requests

`AppRequestUserAttention( NSRequestUserAttentionType type ) = NSInteger`

`AppCancelUserAttentionRequest( NSInteger request )`

#### Presentation options

`AppCurrentSystemPresentationOptions = NSApplicationPresentationOptions`

`AppPresentationOptions = NSApplicationPresentationOptions`

`AppSetPresentationOptions( NSApplicationPresentationOptions options )`

#### Relaunch on login

`AppDisableRelaunchOnLogin`

`AppEnableRelaunchOnLogin`

Occlusion state

`AppOcclusionState = NSApplicationOcclusionState` // macOS 10.9+

Scripting

`AppOrderedDocuments = CFArrayRef` // returns an array of DocumentRefs

`AppOrderedWindows = CFArrayRef` // returns an array of CocoaWindowRefs

Custom

`AppOutputWindow = NSInteger`

`AppSetOutputWindow( NSInteger tag )`

`AppBringAllWindowsToFront`

`AppSetTimer( CFTimeInterval interval, ptr callback, Boolean repeats )`

`AppSetDelegateCallback( ptr callback, ptr userData )`

Convenience

`AppDockTileContentView = ViewRef`

`AppDockTileSize = CGSize`

`AppDockTileShowsApplicationBadge = Boolean`

`AppDockTileSetShowsApplicationBadge( Boolean flag )`

`AppDockTileBadgeLabel = CFStringRef`

`AppDockTileSetBadgeLabel( CFStringRef label )`

`AppDockTileDisplay`

## See Also

[CocoaInit](#)

## Apple documentation

[NSApplication](#)



---

### CocoaInit

statement

#### Description

Use this statement to initiate the Cocoa event loop (as opposed to Carbon event loop).

The **cocoainit** keyword is required for cocoa menus and some widgets. **cocoainit** is also required for 64-bit builds.

#### See Also

[Application](#)



## Array

### Functions

#### Create

```
ArrayWithArray( CFArrayRef array ) = CFArrayRef
ArrayWithObject( CTypeRef obj ) = CFArrayRef
ArrayWithObjects( CTypeRef obj, ... ) = CFArrayRef // a comma-separated list of objects, ending with NULL
```

#### Init

```
ArrayWithContentsOfURL( CFURLRef url ) = CFArrayRef
```

#### Query

```
ArrayContainsObject( CFArrayRef array, CTypeRef obj ) = Boolean
ArrayCount( CFArrayRef array ) = NSUInteger
ArrayFirstObject( CFArrayRef array ) = CTypeRef
ArrayLastObject( CFArrayRef array ) = CTypeRef
ArrayObjectAtIndex( CFArrayRef array, NSUInteger index ) = CTypeRef
ArrayObjectsAtIndexes( CFArrayRef array, IndexSetRef indexes ) = CFArrayRef
ArrayObjectEnumerator( CFArrayRef array ) = EnumeratorRef
ArrayReverseObjectEnumerator( CFArrayRef array ) = EnumeratorRef
```

#### Finding

```
ArrayIndexForObject( CFArrayRef array, CTypeRef obj ) = NSUInteger
ArrayIndexForObjectInRange( CFArrayRef array, CTypeRef obj, CFRange range ) = NSUInteger
ArrayIndexForObjectIdenticalTo( CFArrayRef array, CTypeRef obj ) = NSUInteger
ArrayIndexForObjectPassingTest( CFArrayRef array, ptr testFn ) = NSUInteger
ArrayFirstObjectCommonWithArray( CFArrayRef array1, CFArrayRef array2 ) = CTypeRef
ArrayIsEqualToArray( CFArrayRef array1, CFArrayRef array2 ) = Boolean
ArrayByAddingObject( CFArrayRef array, CTypeRef obj ) = CFArrayRef
ArrayByAddingObjectsFromArray( CFArrayRef array1, CFArrayRef array2 ) = CFArrayRef
ArrayFilteredArrayUsingPredicate( CFArrayRef array, PredicateRef pred ) = CFArrayRef
ArraySubarrayWithRange( CFArrayRef array, CFRange range ) = CFArrayRef
```

#### Sending message to elements

```
ArrayMakeObjectsPerformSelector( CFArrayRef array, CFStringRef selector )
ArrayMakeObjectsPerformSelectorWithObject( CFArrayRef array, CFStringRef selector, CTypeRef object )
ArrayEnumerateObjects( CFArrayRef array, ptr callback, ptr userData )
ArrayEnumerateObjectsWithOptions( CFArrayRef array, NSEnumerationOptions options, ptr callback, ptr userData )
ArrayEnumerateObjectsAtIndexes( CFArrayRef array, IndexSetRef indexSet, NSEnumerationOptions options, ptr callback, ptr userData )
```

#### Comparing

```
ArrayFirstObjectCommonWithArray( CFArrayRef array1, CFArrayRef array2 ) = CTypeRef
ArrayIsEqualToArray( CFArrayRef array1, CFArrayRef array2 ) = Boolean
```

#### Deriving new arrays

```
ArrayByAddingObject( CFArrayRef array, CTypeRef obj ) = CFArrayRef
ArrayByAddingObjectsFromArray( CFArrayRef array1, CFArrayRef array2 ) = CFArrayRef
ArraySubarrayWithRange( CFArrayRef array, CFRange range ) = CFArrayRef
```

#### Sort

```
ArraySortedArrayHint( CFArrayRef array ) = CFDataRef
ArraySortedArrayUsingFunction( CFArrayRef array, ptr comparatorFn, ptr context ) = CFArrayRef
ArraySortedArrayUsingFunctionHint( CFArrayRef array, ptr comparatorFn, ptr context, CFDataRef hint ) = CFArrayRef
ArraySortedArrayUsingDescriptors( CFArrayRef array, CFArrayRef descriptors ) = CFArrayRef
ArraySortedArrayUsingSelector( CFArrayRef array, CFStringRef selector ) = CFArrayRef
```

#### Working with string elements

```
ArrayComponentsJoinedByString( CFArrayRef array, CFStringRef separator ) = CFStringRef
```

#### Description

```
ArrayDescription( CFArrayRef array ) = CFStringRef
ArrayDescriptionWithLocale( CFArrayRef array, CFLocaleRef locale ) = CFStringRef
ArrayDescriptionWithLocaleIndent( CFArrayRef array, CFLocaleRef locale, NSUInteger indentLevel ) = CFStringRef
```

#### Collecting paths

```
ArrayPathsMatchingExtensions( CFArrayRef array, CFArrayRef extensions ) = CFArrayRef
```

#### Key-value coding

```
ArraySetKeyValueForKey( CFArrayRef array, CTypeRef value, CFStringRef key )
ArrayValueForKey( CFArrayRef array, CFStringRef key ) = CTypeRef
```

Shuffle

```
ArrayShuffledArray( CFArrayRef array ) = CFArrayRef// requires GameplayKit framework and macOS 10.12+
```

Instance methods

```
ArrayWriteToURL( CFArrayRef array, CFURLRef url, Boolean atomically ) = Boolean
```

• Mutable array •

Create

```
ArrayWithCapacity( NSUInteger numItems ) = CFMutableArrayRef
```

Adding objects

```
ArrayAddObject( CFMutableArrayRef array, CFTypeRef obj )
```

```
ArrayAddObjectsFromArray( CFMutableArrayRef array, CFArrayRef otherArray )
```

```
ArrayInsertObjectAtIndex( CFMutableArrayRef array, CFTypeRef obj, NSUInteger index )
```

```
ArrayInsertObjectsAtIndexes( CFMutableArrayRef array, CFArrayRef objects, IndexSetRef indexes )
```

Removing objects

```
ArrayRemoveAllObjects( CFMutableArrayRef array )
```

```
ArrayRemoveLastObject( CFMutableArrayRef array )
```

```
ArrayRemoveObject( CFMutableArrayRef array, CFTypeRef obj )
```

```
ArrayRemoveObjectInRange( CFMutableArrayRef array, CFTypeRef obj, CFRange range )
```

```
ArrayRemoveObjectAtIndex( CFMutableArrayRef array, NSUInteger index )
```

```
ArrayRemoveObjectsAtIndexes( CFMutableArrayRef array, IndexSetRef indexes )
```

```
ArrayRemoveObjectsFromArray( CFMutableArrayRef array, CFArrayRef objects )
```

```
ArrayRemoveObjectsInRange( CFMutableArrayRef array, CFRange range )
```

Replacing objects

```
ArrayReplaceObjectAtIndex( CFMutableArrayRef array, CFTypeRef obj, NSUInteger index )
```

```
ArrayReplaceObjectsAtIndexes( CFMutableArrayRef array, CFArrayRef objects, IndexSetRef indexes )
```

```
ArraySetArray( CFMutableArrayRef array, CFArrayRef otherArray )
```

Filter content

```
ArrayFilterUsingPredicate( CFMutableArrayRef array, PredicateRef predicate )
```

Rearranging objects

```
ArrayExchangeObjects( CFMutableArrayRef array, NSUInteger index1, NSUInteger index2 )
```

```
ArraySortUsingDescriptors( CFMutableArrayRef array, CFArrayRef descriptors )
```

```
ArraySortUsingFunction( CFMutableArrayRef array, ptr comparator, ptr context )
```

```
ArraySortUsingSelector( CFMutableArrayRef array, CFStringRef selector )
```

## Apple documentation

[NSArray](#)

[NSMutableArray](#)



---

### Asset

#### Functions

Create

```
AssetWithURL( CFURLRef url ) = AssetRef
```

Inspect

```
AssetDuration( AssetRef ref ) = CMTime
```

Tracks

```
AssetTracks( AssetRef ref ) = CFArrayRef
```

```
AssetTrackWithID( AssetRef ref, CMPersistentTrackID trackID ) = AssetTrackRef
```

```
AssetTracksWithMediaType( AssetRef ref, CFStringRef type ) = CFArrayRef
```

#### Apple documentation

[AVAsset](#)



---

## AssetTrack

### Functions

Info

```
AssetTrackAsset( AssetTrackRef ref ) = AssetRef  
AssetTrackID( AssetTrackRef ref ) = CMPersistentTrackID  
AssetTrackMediaType( AssetTrackRef ref ) = CFStringRef  
AssetTrackTotalSampleDataLength( AssetTrackRef ref ) = SInt64
```

### Apple documentation

[AVAssetTrack](#)



## AttributedString

### Functions

#### Create

```
AttributedStringWithData( CFDataRef dta, CFDictionaryRef options, CFDictionaryRef *attributes, ErrorRef *err ) = CFAttributedStringRef
AttributedStringWithDocFormat( CFDataRef dta, CFDictionaryRef *attributes ) = CFAttributedStringRef
AttributedStringWithHTML( CFDataRef dta, CFDictionaryRef *attributes ) = CFAttributedStringRef
AttributedStringWithHTMLBaseURL( CFDataRef dta, CFURLRef url, CFDictionaryRef *attributes ) = CFAttributedStringRef
AttributedStringWithHTMLOptions( CFDataRef dta, CFDictionaryRef options, CFDictionaryRef *attributes ) = CFAttributedStringRef
AttributedStringWithRTF( CFDataRef dta, CFDictionaryRef *attributes ) = CFAttributedStringRef
AttributedStringWithRTFD( CFDataRef dta, CFDictionaryRef *attributes ) = CFAttributedStringRef
AttributedStringWithRTFDFileWrapper( FileWrapperRef fw, CFDictionaryRef *attributes ) = CFAttributedStringRef
AttributedStringWithURL( CFURLRef url, CFDictionaryRef options, CFDictionaryRef *attributes, ErrorRef *err ) = CFAttributedStringRef
AttributedStringWithAttachment( TextAttachmentRef attachment ) = CFAttributedStringRef
```

#### Character info

```
AttributedStringString( CFAttributedStringRef aString ) = CFStringRef
AttributedStringLength( CFAttributedStringRef aString ) = NSUInteger
```

#### Attribute info

```
AttributedStringAttributesAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = CFDictionaryRef
AttributedStringAttributesAtIndexInRange( CFAttributedStringRef aString, NSUInteger index, CFRange *longestEffectiveRange, CFRange range ) = CFDictionaryRef
AttributedStringAttributeAtIndex( CFAttributedStringRef aString, CFStringRef attrName, NSUInteger index, CFRange *longestEffectiveRange ) = CFTypeRef
AttributedStringAttributeAtIndexInRange( CFAttributedStringRef aString, CFStringRef attrName, NSUInteger index, CFRange *longestEffectiveRange, CFRange range ) = CFTypeRef
```

#### Compare

```
AttributedStringIsEqual( CFAttributedStringRef aString1, CFAttributedStringRef aString2 ) = Boolean
```

#### Substring

```
AttributedStringAttributedStringFromRange( CFAttributedStringRef aString, CFRange range ) = CFAttributedStringRef
```

#### Enumerate

```
AttributedStringEnumerateAttributeInRange( CFAttributedStringRef aString, CFStringRef attrName, CFRange range, NSAttributedStringEnumerationOptions options, ptr fnAddress, ptr userData )
AttributedStringEnumerateAttributesInRange( CFAttributedStringRef aString, CFRange range, NSAttributedStringEnumerationOptions options, ptr fnAddress, ptr userData )
```

#### Font info

```
AttributedStringFontAttributesInRange( CFAttributedStringRef aString, CFRange range ) = CFDictionaryRef
AttributedStringRulerAttributesInRange( CFAttributedStringRef aString, CFRange range ) = CFDictionaryRef
```

#### Linguistic units

```
AttributedStringDoubleClickAtIndex( CFAttributedStringRef aString, NSUInteger index ) = CFRange
AttributedStringLineBreakBeforeIndex( CFAttributedStringRef aString, NSUInteger index, CFRange withinRange ) = NSUInteger
AttributedStringLineBreakByHyphenatingBeforeIndex( CFAttributedStringRef aString, NSUInteger index, CFRange withinRange ) = NSUInteger
AttributedStringNextWordFromIndex( CFAttributedStringRef aString, NSUInteger index, Boolean forward ) = NSUInteger
```

#### Ranges

```
AttributedStringItemNumberInTextList( CFAttributedStringRef aString, TextListRef list, NSUInteger index ) = NSInteger
AttributedStringRangeOfTextBlock( CFAttributedStringRef aString, TextBlockRef block, NSUInteger index ) = CFRange
AttributedStringRangeOfTextList( CFAttributedStringRef aString, TextListRef list, NSUInteger index ) = CFRange
AttributedStringRangeOfTextTable( CFAttributedStringRef aString, TextTableRef table, NSUInteger index ) = CFRange
```

#### Data

```
AttributedStringDataFromRange( CFAttributedStringRef aString, CFRange range, CFDictionaryRef attributes, ErrorRef *err ) = CFDataRef
AttributedStringFileWrapperFromRange( CFAttributedStringRef aString, CFRange range, CFDictionaryRef attributes, ErrorRef *err ) = FileWrapperRef
AttributedStringDocFormatFromRange( CFAttributedStringRef aString, CFRange range, CFDictionaryRef attributes ) = CFDataRef
AttributedStringRTFFromRange( CFAttributedStringRef aString, CFRange range, CFDictionaryRef attributes ) = CFDataRef
```



```
AttributedStringRTFDFFromRange( CFAttributedStringRef aString, CFRange range, CFDictionaryRef attributes ) = CFDataRef
AttributedStringRTFDFFileWrapperFromRange( CFAttributedStringRef aString, CFRange range, CFDictionaryRef attributes ) = FileWrapperRef
```

#### Draw

```
AttributedStringDrawAtPoint( CFAttributedStringRef aString, CGPoint pt )
AttributedStringDrawInRect( CFAttributedStringRef aString, CGRect r )
AttributedStringDrawWithRect( CFAttributedStringRef aString, CGRect r, NSStringDrawingOptions options, StringDrawingContextRef ctx )// macOS 10.11+
```

#### Metrics

```
AttributedStringSize( CFAttributedStringRef aString ) = CGSize
AttributedStringBoundingRectWithSize( CFAttributedStringRef aString, CGSize size, NSStringDrawingOptions options, StringDrawingContextRef ctx ) = CGRect// macOS 10.11+
AttributedStringContainsAttachmentsInRange( CFAttributedStringRef aString, CFRange range ) = Boolean
```

#### Data sources

```
AttributedStringTextTypes = CFArrayRef
AttributedStringTextUnfilteredTypes = CFArrayRef
```

#### - Mutable attributed string -

##### Create

```
AttributedStringWithString( CFStringRef string ) = CFMutableAttributedStringRef
AttributedStringWithAttributes( CFStringRef string, CFDictionaryRef attributes ) = CFMutableAttributedStringRef
```

##### Character info

```
AttributedStringMutableString( CFMutableAttributedStringRef aString ) = CFMutableStringRef
```

#### Change characters

```
AttributedStringReplaceCharactersInRange( CFMutableAttributedStringRef aString, CFStringRef string, CFRange range )
AttributedStringDeleteCharactersInRange( CFMutableAttributedStringRef aString, CFRange range )
```

#### Change attributes

```
AttributedStringSetAttributesInRange( CFMutableAttributedStringRef aString, CFDictionaryRef attrs, CFRange range )
AttributedStringAddAttributeInRange( CFMutableAttributedStringRef aString, CFStringRef name, CFTypeRef value, CFRange range )
AttributedStringAddAttributesInRange( CFMutableAttributedStringRef aString, CFDictionaryRef attrs, CFRange range )
AttributedStringRemoveAttributeInRange( CFMutableAttributedStringRef aString, CFStringRef name, CFRange range )
AttributedStringApplyFontTraitsInRange( CFMutableAttributedStringRef aString, NSFontTraitMask traitMask, CFRange range )
AttributedStringSetAlignmentInRange( CFMutableAttributedStringRef aString, NSTextAlignment alignment, CFRange range )
AttributedStringSetBaseWritingDirectionInRange( CFMutableAttributedStringRef aString, NSWritingDirection value, CFRange range )
AttributedStringSubscriptRange( CFMutableAttributedStringRef aString, CFRange range )
AttributedStringSuperscriptRange( CFMutableAttributedStringRef aString, CFRange range )
AttributedStringUnscriptRange( CFMutableAttributedStringRef aString, CFRange range )
```

#### Change characters and attributes

```
AttributedStringAppendAttributedString( CFMutableAttributedStringRef aString, CFAttributedStringRef attrString )
AttributedStringInsertAttributedStringAtIndex( CFMutableAttributedStringRef aString, CFAttributedStringRef attrString, NSUInteger index )
AttributedStringReplaceCharactersInRangeWithAttributedString( CFMutableAttributedStringRef aString, CFRange range, CFAttributedStringRef attrString )
AttributedStringSetAttributedString( CFMutableAttributedStringRef aString, CFAttributedStringRef attrString )
```

#### Grouping changes

```
AttributedStringBeginEditing( CFMutableAttributedStringRef aString )
AttributedStringEndEditing( CFMutableAttributedStringRef aString )
```

#### Fix attributes after changes

```
AttributedStringFixAttributesInRange( CFMutableAttributedStringRef aString, CFRange range )
AttributedStringFixAttachmentAttributeInRange( CFMutableAttributedStringRef aString, CFRange range )
AttributedStringFixFontAttributeInRange( CFMutableAttributedStringRef aString, CFRange range )
AttributedStringFixParagraphStyleAttributeInRange( CFMutableAttributedStringRef aString, CFRange range )
```

#### Read content

```
AttributedStringReadFromData( CFMutableAttributedStringRef aString, CFDataRef dta, CFDictionaryRef options, CFDictionaryRef *docAttributes, NSError *err ) = Boolean
AttributedStringReadFromURL( CFMutableAttributedStringRef aString, CFURLRef url, CFDictionaryRef options, CFDictionaryRef *docAttributes, NSError *err ) = Boolean
```

#### - Convenience -

##### String

```
AttributedStringSetString( CFMutableAttributedStringRef aString, CFStringRef string )
```

##### Append

```
AttributedStringAppendCharacters( CFMutableAttributedStringRef aString, CFStringRef string )
```

## Font

```
AttributedStringFontAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = FontRef
AttributedStringSetFontWithNameInRange( CFMutableAttributedStringRef aString, CFStringRef name, CGFloat size,
CFRange range )
AttributedStringSetFontWithName( CFMutableAttributedStringRef aString, CFStringRef name, CGFloat size )
```

## Alignment

```
AttributedStringSetAlignment( CFMutableAttributedStringRef aString, NSTextAlignment alignment )
```

## Stroke width

```
AttributedStringStrokeWidthAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) =
CGFloat
AttributedStringSetStrokeWidthInRange( CFMutableAttributedStringRef aString, CGFloat width, CFRange range )
AttributedStringSetStrokeWidth( CFMutableAttributedStringRef aString, CGFloat width )
```

## Stroke color

```
AttributedStringStrokeColorAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) =
ColorRef
AttributedStringSetStrokeColorInRange( CFMutableAttributedStringRef aString, ColorRef col, CFRange range )
AttributedStringSetStrokeColor( CFMutableAttributedStringRef aString, ColorRef col )
```

## Foreground color

```
AttributedStringForegroundColorAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) =
ColorRef
AttributedStringSetForegroundColorInRange( CFMutableAttributedStringRef aString, ColorRef col, CFRange range )
AttributedStringSetForegroundColor( CFMutableAttributedStringRef aString, ColorRef col )
```

## Background color

```
AttributedStringBackgroundColorAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) =
ColorRef
AttributedStringSetBackgroundColorInRange( CFMutableAttributedStringRef aString, ColorRef col, CFRange range )
AttributedStringSetBackgroundColor( CFMutableAttributedStringRef aString, ColorRef col )
```

## Shadow

```
AttributedStringShadowAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) =
ShadowRef
AttributedStringSetShadowInRange( CFMutableAttributedStringRef aString, CGSize offset, CGFloat blur, CFRange range )
AttributedStringSetShadow( CFMutableAttributedStringRef aString, CGSize offset, CGFloat blur )
```

## Shadow color

```
AttributedStringShadowColorAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) =
ColorRef
AttributedStringSetShadowColorInRange( CFMutableAttributedStringRef aString, ColorRef col, CFRange range )
AttributedStringSetShadowColor( CFMutableAttributedStringRef aString, ColorRef col )
```

## Underline

```
AttributedStringUnderlineAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) =
NSInteger
AttributedStringSetUnderlineInRange( CFMutableAttributedStringRef aString, NSInteger style, CFRange range )
AttributedStringSetUnderline( CFMutableAttributedStringRef aString, NSUnderlineStyle style )
```

## Underline color

```
AttributedStringUnderlineColorAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange )=
ColorRef
AttributedStringSetUnderlineColorInRange( CFMutableAttributedStringRef aString, ColorRef col, CFRange range )
AttributedStringSetUnderlineColor( CFMutableAttributedStringRef aString, ColorRef col )
```

## Strikethrough

```
AttributedStringStrikethroughColorAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange
*effectiveRange ) = ColorRef
AttributedStringSetStrikethroughInRange( CFMutableAttributedStringRef aString, NSInteger style, CFRange range )
AttributedStringSetStrikethrough( CFMutableAttributedStringRef aString, NSInteger style )
```

## Strikethrough color

```
AttributedStringStrikethroughAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) =
NSInteger
AttributedStringSetStrikethroughColorInRange( CFMutableAttributedStringRef aString, ColorRef col, CFRange range )
AttributedStringSetStrikethroughColor( CFMutableAttributedStringRef aString, ColorRef col )
```

## Superscript

```
AttributedStringSuperscriptAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) =
NSInteger
AttributedStringSetSuperscriptInRange( CFMutableAttributedStringRef aString, NSInteger value, CFRange range )
AttributedStringSetSuperscript( CFMutableAttributedStringRef aString, NSInteger value )
```

## Expansion

```
AttributedStringExpansionAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = float
AttributedStringSetExpansionInRange( CFMutableAttributedStringRef aString, float value, CFRange range )
AttributedStringSetExpansion( CFMutableAttributedStringRef aString, float value )
```

## Obliqueness

```
AttributedStringObliquenessAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = float
AttributedStringSetObliquenessInRange( CFMutableAttributedStringRef aString, float value, CFRange range )
AttributedStringSetObliqueness( CFMutableAttributedStringRef aString, float value )
```

## Link

```
AttributedStringLinkAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = CFTyperef
AttributedStringSetLinkInRange( CFMutableAttributedStringRef aString, CFTyperef value, CFRange range )
AttributedStringSetLink( CFMutableAttributedStringRef aString, CFTyperef value )
```

## Baseline offset

```
AttributedStringBaselineOffsetAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = float
AttributedStringSetBaselineOffsetInRange( CFMutableAttributedStringRef aString, float value, CFRange range )
AttributedStringSetBaselineOffset( CFMutableAttributedStringRef aString, float value )
```

## Kerning

```
AttributedStringKerningAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = CGFloat
AttributedStringSetKerningInRange( CFMutableAttributedStringRef aString, CGFloat value, CFRange range )
AttributedStringSetKerning( CFMutableAttributedStringRef aString, CGFloat value )
```

## - paragraph styles -

### Alignment

```
AttributedStringParagraphAlignmentAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = NSTextAlignment
AttributedStringSetParagraphAlignmentInRange( CFMutableAttributedStringRef aString, NSTextAlignment value, CFRange range )
AttributedStringSetParagraphAlignment( CFMutableAttributedStringRef aString, NSTextAlignment value )
```

### First line head indent

```
AttributedStringFirstLineHeadIndentAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = CGFloat
AttributedStringSetFirstLineHeadIndentInRange( CFMutableAttributedStringRef aString, CGFloat value, CFRange range )
AttributedStringSetFirstLineHeadIndent( CFMutableAttributedStringRef aString, CGFloat value )
```

### Head indent

```
AttributedStringHeadIndentAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = CGFloat
AttributedStringSetHeadIndentInRange( CFMutableAttributedStringRef aString, CGFloat value, CFRange range )
AttributedStringSetHeadIndent( CFMutableAttributedStringRef aString, CGFloat value )
```

### Tail indent

```
AttributedStringTailIndentAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = CGFloat
AttributedStringSetTailIndentInRange( CFMutableAttributedStringRef aString, CGFloat value, CFRange range )
AttributedStringSetTailIndent( CFMutableAttributedStringRef aString, CGFloat value )
```

### Line break mode

```
AttributedStringLineBreakModeAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = NSLineBreakMode
AttributedStringSetLineBreakModeInRange( CFMutableAttributedStringRef aString, NSLineBreakMode value, CFRange range )
AttributedStringSetLineBreakMode( CFMutableAttributedStringRef aString, NSLineBreakMode value )
```

### Maximum line height

```
AttributedStringMaximumLineHeightAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = CGFloat
AttributedStringSetMaximumLineHeightInRange( CFMutableAttributedStringRef aString, CGFloat value, CFRange range )
AttributedStringSetMaximumLineHeight( CFMutableAttributedStringRef aString, CGFloat value )
```

### Minimum line height

```
AttributedStringMinimumLineHeightAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = CGFloat
AttributedStringSetMinimumLineHeightInRange( CFMutableAttributedStringRef aString, CGFloat value, CFRange range )
AttributedStringSetMinimumLineHeight( CFMutableAttributedStringRef aString, CGFloat value )
```

### Line spacing

```
AttributedStringLineSpacingAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = CGFloat
AttributedStringSetLineSpacingInRange( CFMutableAttributedStringRef aString, CGFloat value, CFRange range )
AttributedStringSetLineSpacing( CFMutableAttributedStringRef aString, CGFloat value )
```

### Paragraph spacing

```
AttributedStringParagraphSpacingAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) = CGFloat
AttributedStringSetParagraphSpacingInRange( CFMutableAttributedStringRef aString, CGFloat value, CFRange range )
AttributedStringSetParagraphSpacing( CFMutableAttributedStringRef aString, CGFloat value )
```

#### Paragraph spacing before

```
AttributedStringParagraphSpacingBeforeAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange
*effectiveRange ) = CGFloat
AttributedStringSetParagraphSpacingBeforeInRange( CFMutableAttributedStringRef aString, CGFloat value, CFRange range
)
AttributedStringSetParagraphSpacingBefore( CFMutableAttributedStringRef aString, CGFloat value )
```

#### Base writing direction

```
AttributedStringSetBaseWritingDirection( CFMutableAttributedStringRef aString, NSWritingDirection value )
```

#### Line height multiple

```
AttributedStringLineHeightMultipleAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange
*effectiveRange ) = CGFloat
AttributedStringSetLineHeightMultipleInRange( CFMutableAttributedStringRef aString, CGFloat value, CFRange range )
AttributedStringSetLineHeightMultiple( CFMutableAttributedStringRef aString, CGFloat value )
```

#### Add tab stop

```
AttributedStringAddTabStopInRange( CFMutableAttributedStringRef aString, NSTextAlignment alignment, CGFloat
location, CFDictionaryRef options, CFRange range )
AttributedStringAddTabStop( CFMutableAttributedStringRef aString, NSTextAlignment alignment, CGFloat location,
CFDictionaryRef options )
```

#### Set tab stops

```
AttributedStringTabStopsAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) =
CFArrayRef
AttributedStringSetTabStopsInRange( CFMutableAttributedStringRef aString, CFArrayRef tabStops, CFRange range )
AttributedStringSetTabStops( CFMutableAttributedStringRef aString, CFArrayRef tabStops )
```

#### Set default tab interval

```
AttributedStringDefaultTabIntervalAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange
*effectiveRange ) = CGFloat
AttributedStringSetDefaultTabIntervalInRange( CFMutableAttributedStringRef aString, CGFloat interval, CFRange
range )
AttributedStringSetDefaultTabInterval( CFMutableAttributedStringRef aString, CGFloat interval )
```

#### Header level

```
AttributedStringHeaderLevelAtIndex( CFAttributedStringRef aString, NSUInteger index, CFRange *effectiveRange ) =
NSInteger
AttributedStringSetHeaderLevelInRange( CFMutableAttributedStringRef aString, NSInteger value, CFRange range )
AttributedStringSetHeaderLevel( CFMutableAttributedStringRef aString, NSInteger value )
```

### Apple documentation

[NSAttributedString](#)

[NSMutableAttributedString](#)



---

## AVPlayer

### Description

For use with AVPlayerView.

### Functions

Create

```
AVPlayerWithURL( CFURLRef url ) = AVPlayerRef
AVPlayerWithPlayerItem( AVPlayerItemRef item ) = AVPlayerRef
```

Playback

```
AVPlayerPlay( AVPlayerRef player )
AVPlayerPause( AVPlayerRef player )
AVPlayerRate( AVPlayerRef player ) = float
AVPlayerSetRate( AVPlayerRef player, float rate )
AVPlayerActionAtItemEnd( AVPlayerRef player ) = AVPlayerActionAtItemEnd
AVPlayerSetActionAtItemEnd( AVPlayerRef player, AVPlayerActionAtItemEnd action )
AVPlayerReplaceCurrentItemWithPlayerItem( AVPlayerRef player, AVPlayerItemRef item )
```

Waiting behavior

```
AVPlayerAutomaticallyWaitsToMinimizeStalling( AVPlayerRef player ) = Boolean// macOS 10.12+
AVPlayerSetAutomaticallyWaitsToMinimizeStalling( AVPlayerRef player, Boolean flag )// macOS 10.12+
AVPlayerReasonForWaitingToPlay( AVPlayerRef player ) = CFStringRef// macOS 10.12+
AVPlayerTimeControlStatus( AVPlayerRef player ) = AVPlayerTimeControlStatus// macOS 10.12+
AVPlayerPlayImmediatelyAtRate( AVPlayerRef player, float rate )// macOS 10.12+
```

Managing time

```
AVPlayerCurrentTime( AVPlayerRef player ) = CMTime
AVPlayerSeekToTime( AVPlayerRef player, CMTime tm )
AVPlayerSeekToDate( AVPlayerRef player, CFDateRef dt )
AVPlayerSeekToTimeWithTolerance( AVPlayerRef player, CMTime tm, CMTime toleranceBefore, CMTime toleranceAfter )
```

Audio output

```
AVPlayerIsMuted( AVPlayerRef player ) = Boolean
AVPlayerSetMuted( AVPlayerRef player, Boolean flag )
AVPlayerVolume( AVPlayerRef player ) = float
AVPlayerSetVolume( AVPlayerRef player, float volume )
```

Player properties

```
AVPlayerStatus( AVPlayerRef player ) = AVPlayerStatus
AVPlayerError( AVPlayerRef player ) = ErrorRef
AVPlayerCurrentItem( AVPlayerRef player ) = AVPlayerItemRef
```

### Apple documentation

[AVPlayer](#)



---

## AVPlayerItem

### Functions

Create

```
AVPlayerItemWithURL( CFURLRef url ) = AVPlayerItemRef
```

Inspecting

```
AVPlayerItemAsset( AVPlayerItemRef item ) = AssetRef  
AVPlayerItemStatus( AVPlayerItemRef item ) = AVPlayerItemStatus  
AVPlayerItemDuration( AVPlayerItemRef item ) = CMTime  
AVPlayerItemTimebase( AVPlayerItemRef item ) = CMTimebaseRef  
AVPlayerItemPresentationSize( AVPlayerItemRef item ) = CGSize
```

Moving the playhead

```
AVPlayerItemStepByCount( AVPlayerItemRef item, NSInteger count )
```

Playback info

```
AVPlayerItemCanPlayReverse( AVPlayerItemRef item ) = Boolean  
AVPlayerItemCanPlayFastForward( AVPlayerItemRef item ) = Boolean  
AVPlayerItemCanPlayFastReverse( AVPlayerItemRef item ) = Boolean  
AVPlayerItemCanPlaySlowForward( AVPlayerItemRef item ) = Boolean  
AVPlayerItemCanPlaySlowReverse( AVPlayerItemRef item ) = Boolean  
AVPlayerItemCanStepBackward( AVPlayerItemRef item ) = Boolean  
AVPlayerItemCanStepForward( AVPlayerItemRef item ) = Boolean
```

Timing info

```
AVPlayerItemCurrentTime( AVPlayerItemRef item ) = CMTime  
AVPlayerItemCurrentDate( AVPlayerItemRef item ) = CFDateRef  
AVPlayerItemForwardPlaybackEndTime( AVPlayerItemRef item ) = CMTime  
AVPlayerItemReversePlaybackEndTime( AVPlayerItemRef item ) = CMTime
```

### Apple documentation

[AVPlayerItem](#)



---

## AVPlayerLayer

### Functions

Create

```
AVPlayerLayerWithPlayer( PlayerRef player ) = AVPlayerLayerRef
```

Configure

```
AVPlayerLayerPlayer( AVPlayerLayerRef ref ) = PlayerRef
```

```
AVPlayerLayerIsReadyForDisplay( AVPlayerLayerRef ref ) = Boolean
```

```
AVPlayerLayerVideoGravity( AVPlayerLayerRef ref ) = CFStringRef
```

```
AVPlayerLayerVideoRect( AVPlayerLayerRef ref ) = CGRect
```

```
AVPlayerLayerPixelBufferAttributes( AVPlayerLayerRef ref ) = CFDictionaryRef
```

```
AVMakeRectWithAspectRatioInsideRect( CGSize aspectRatio, CGRect boundingRect ) = CGRect
```

### Apple documentation

[AVPlayerLayer](#)



## AVPlayerView

### Syntax

**avplayerview** *tag*, *rect*, *controlsStyle*, *wndTag*

### Parameters

<i>tag</i>	A number to identify the webview. A negative tag hides the view.
<i>rect</i>	Origin and size of the view in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>controlsStyle</i>	The player view's controls style.
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Description

The **avplayerview** statement puts a new avplayerview in the current output cocoa window, or alters an existing avplayerview characteristics.

### Functions

```
AVPlayerViewWithTag( NSInteger tag ) = AVPlayerViewRef
```

Playback view

```
AVPlayerViewPlayer( NSInteger tag ) = PlayerRef
AVPlayerViewSetPlayer( NSInteger tag, PlayerRef player )
AVPlayerViewSetPlayer( NSInteger tag, PlayerRef player )
```

Customization

```
AVPlayerViewControlsStyle( NSInteger tag ) = AVPlayerViewControlsStyle
AVPlayerViewSetControlsStyle( NSInteger tag, AVPlayerViewControlsStyle style )
AVPlayerViewShowsFrameSteppingButtons( NSInteger tag ) = Boolean
AVPlayerViewSetShowsFrameSteppingButtons( NSInteger tag, Boolean flag )
AVPlayerViewShowsSharingServiceButton( NSInteger tag ) = Boolean
AVPlayerViewSetShowsSharingServiceButton( NSInteger tag, Boolean flag )
AVPlayerViewShowsFullScreenToggleButton( NSInteger tag ) = Boolean
AVPlayerViewSetShowsFullScreenToggleButton( NSInteger tag, Boolean flag )
AVPlayerViewContentOverlayView( NSInteger tag ) = ViewRef
AVPlayerViewActionPopUpButtonMenu( NSInteger tag ) = CocoaMenuRef
AVPlayerViewSetActionPopUpButtonMenu( NSInteger tag, CocoaMenuRef m )
AVPlayerViewUpdatesNowPlayingInfoCenter( NSInteger tag ) = Boolean// macOS 10.13+
AVPlayerViewSetUpdatesNowPlayingInfoCenter( NSInteger tag, Boolean flag )// macOS 10.13+
```

Configure video

```
AVPlayerViewIsReadyForDisplay( NSInteger tag ) = Boolean
AVPlayerViewVideoBounds( NSInteger tag ) = CGRect
AVPlayerViewVideoGravity( NSInteger tag ) = CFStringRef
AVPlayerViewSetVideoGravity( NSInteger tag, CFStringRef gravity )
```

Trimming playback content

```
AVPlayerViewCanBeginTrimming( NSInteger tag ) = Boolean
```

Display chapter and title

```
AVPlayerViewFlashChapterNumber( NSInteger tag, NSUInteger number, CFStringRef title )
```

### Apple documentation

[AVPlayerView](#)





## BezierPath

### Functions

create

```
BezierPathInit = BezierPathRef// autoreleased
BezierPathWithOvalInRect( CGRect r ) = BezierPathRef
BezierPathWithRect( CGRect r ) = BezierPathRef
BezierPathWithRoundedRect( CGRect r, CGFloat xRadius, CGFloat yRadius ) = BezierPathRef
BezierPathByFlatteningPath( BezierPathRef path ) = BezierPathRef
BezierPathByReversingPath( BezierPathRef path ) = BezierPathRef
```

Constructing paths

```
BezierPathMoveToPoint( BezierPathRef path, CGPoint pt )
BezierPathLineToPoint( BezierPathRef path, CGPoint pt )
BezierPathCurveToPoint( BezierPathRef path, CGPoint pt, CGPoint cp1, CGPoint cp2 )
BezierPathClose( BezierPathRef path )
BezierPathRelativeMoveToPoint( BezierPathRef path, CGPoint pt )
BezierPathRelativeLineToPoint( BezierPathRef path, CGPoint pt )
BezierPathRelativeCurveToPoint( BezierPathRef path, CGPoint pt, CGPoint cp1, CGPoint cp2 )
```

Appending

```
BezierPathAppendPath( BezierPathRef path1, BezierPathRef path2 )
BezierPathAppendPathWithPoints( BezierPathRef path, CGPoint *points, NSInteger count )
BezierPathAppendPathWithOvalInRect( BezierPathRef path, CGRect r )
BezierPathAppendPathWithArcFromPoint( BezierPathRef path, CGPoint pt1, CGPoint pt2, CGFloat radius )
BezierPathAppendPathWithArcWithCenter( BezierPathRef path, CGPoint center, CGFloat radius, CGFloat startAngle,
CGFloat endAngle, Boolean clockwise )
```

```
BezierPathAppendPathWithRect( BezierPathRef path, CGRect r )
BezierPathAppendPathWithRoundedRect( BezierPathRef path, CGRect r, CGFloat xRadius, CGFloat yRadius )
```

Attributes

```
BezierPathSetWindingRule( BezierPathRef path, NSWindingRule windingRule )
BezierPathSetLineCapStyle( BezierPathRef path, NSLineCapStyle style )
BezierPathSetLineJoinStyle( BezierPathRef path, NSLineJoinStyle style )
BezierPathSetLineWidth( BezierPathRef path, CGFloat lineWidth )
BezierPathSetMiterLimit( BezierPathRef path, CGFloat limit )
BezierPathSetFlatness( BezierPathRef path, CGFloat limit )
BezierPathGetLineDash( BezierPathRef path, CGFloat *pattern, NSInteger *count, CGFloat *phase )
BezierPathSetLineDash( BezierPathRef path, CGFloat *pattern, NSInteger count, CGFloat phase )
```

Default attributes

```
BezierPathDefaultWindingRule = NSWindingRule
BezierPathSetDefaultWindingRule( NSWindingRule rule )
BezierPathDefaultLineCapStyle = NSLineCapStyle
BezierPathSetDefaultLineCapStyle( NSLineCapStyle style )
BezierPathDefaultLineJoinStyle = NSLineJoinStyle
BezierPathSetDefaultLineJoinStyle( NSLineJoinStyle style )
BezierPathDefaultLineWidth = CGFloat
BezierPathSetDefaultLineWidth( CGFloat lineWidth )
BezierPathDefaultMiterLimit = CGFloat
BezierPathSetDefaultMiterLimit( NSWindingRule limit )
BezierPathDefaultFlatness = CGFloat
BezierPathSetDefaultFlatness( NSWindingRule flatness )
```

Drawing paths

```
BezierPathStroke( BezierPathRef path )
BezierPathFill( BezierPathRef path )
```

```
BezierPathAddClip( BezierPathRef path )
BezierPathSetClip( BezierPathRef path )
BezierPathClipRect( CGRect r )
```

Hit detection

```
BezierPathContainsPoint( BezierPathRef path, CGPoint pt ) = Boolean
```

Query

```
BezierPathBounds( BezierPathRef path ) = CGRect
BezierPathControlPointBounds( BezierPathRef path ) = CGRect
BezierPathCurrentPoint( BezierPathRef path ) = CGPoint
BezierPathIsEmpty( BezierPathRef path ) = Boolean
```

Apply transformation

```
BezierPathTransformUsingAffineTranform( BezierPathRef path, AffineTransformRef tx )
```

Elements

```
BezierPathElementCount( BezierPathRef path ) = NSInteger
```

```
BezierPathElementAtIndex( BezierPathRef path, NSInteger index ) = NSBezierPathElement
```

```
BezierPathElementAtIndexAssociatedPoints( BezierPathRef path, NSInteger index, CGPoint *points ) =  
NSBezierPathElement
```

```
BezierPathRemoveAllPoints( BezierPathRef path )
```

```
BezierPathSetAssociatedPointsAtIndex( BezierPathRef path, CGPoint *points, NSInteger index )
```

- convenience -

```
BezierPathStrokeFillOvalInRect( CGRect r, CGFloat lineWidth, ColorRef strokeCol, ColorRef fillCol )
```

```
BezierPathStrokeFillRect( CGRect r, CGFloat lineWidth, ColorRef strokeCol, ColorRef fillCol )
```

```
BezierPathStrokeFillRoundedRect( CGRect r, CGFloat xRadius, CGFloat yRadius, CGFloat lineWidth, ColorRef strokeCol,  
ColorRef fillCol )
```

```
BezierPathStrokeFillPolygon( CFArrayRef pts, CGFloat lineWidth, ColorRef strokeCol, ColorRef fillCol )
```

```
BezierPathStrokeOvalInRect( CGRect r, CGFloat lineWidth, ColorRef strokeCol )
```

```
BezierPathStrokeRect( CGRect r, CGFloat lineWidth, ColorRef strokeCol )
```

```
BezierPathStrokeRoundedRect( CGRect r, CGFloat xRadius, CGFloat yRadius, CGFloat lineWidth, ColorRef strokeCol )
```

```
BezierPathStrokeRotatedRect( CGRect r, CGFloat x, CGFloat y, CGFloat angle, CGFloat lineWidth, ColorRef strokeCol )
```

```
BezierPathStrokeLine( CGPoint pt1, CGPoint pt2, CGFloat lineWidth, ColorRef strokeCol )
```

```
BezierPathStrokeCurve( CGPoint pt1, CGPoint pt2, CGPoint cpl, CGPoint cp2, CGFloat lineWidth, ColorRef strokeCol )
```

```
BezierPathStrokePolygon( CFArrayRef pts, CGFloat lineWidth, ColorRef strokeCol )
```

```
BezierPathFillOvalInRect( CGRect r, ColorRef fillCol )
```

```
BezierPathFillRect( CGRect r, ColorRef fillCol )
```

```
BezierPathFillRoundedRect( CGRect r, CGFloat xRadius, CGFloat yRadius, ColorRef fillCol )
```

```
BezierPathFillPolygon( CFArrayRef pts, ColorRef fillCol )
```

## Apple documentation

[NSBezierPath](#)



## BitmapImageRep

### Functions

#### Create

```
BitmapImageRepWithData( CFDataRef dta ) = BitmapImageRepRef
BitmapImageRepsWithData( CFDataRef dta ) = CFArrayRef
BitmapImageRepColorizeByMapping( BitmapImageRepRef rep, CGFloat midPt, ColorRef midPtCol, ColorRef shadowCol, ColorRef lightCol )
BitmapImageRepWithDataPlanes( ptr planes, NSInteger wide, NSInteger high, NSInteger bps, NSInteger spp, Boolean hasAlpha, Boolean isPlanar, CFStringRef csName, NSInteger rBytes, NSInteger pBits ) = BitmapImageRepRef
BitmapImageRepWithCGImage( CGImageRef ref ) = BitmapImageRepRef
BitmapImageRepWithCIImage( CIImageRef ref ) = BitmapImageRepRef
BitmapImageRepWithFocusedViewRect( CGRect r ) = BitmapImageRepRef
BitmapImageRepForIncrementalLoad = BitmapImageRepRef
```

#### Image info

```
BitmapImageRepFormat( BitmapImageRepRef ref ) = NSBitmapFormat
BitmapImageRepBitsPerPixel( BitmapImageRepRef ref ) = NSInteger
BitmapImageRepBytesPerPlane( BitmapImageRepRef ref ) = NSInteger
BitmapImageRepBytesPerRow( BitmapImageRepRef ref ) = NSInteger
BitmapImageRepIsPlanar( BitmapImageRepRef ref ) = Boolean
BitmapImageRepNumberOfPlanes( BitmapImageRepRef ref ) = NSInteger
BitmapImageRepSamplesPerPixel( BitmapImageRepRef ref ) = NSInteger
```

#### Image data

```
BitmapImageRepBitmapData( BitmapImageRepRef rep ) = ptr
```

#### Image representation

```
BitmapImageRepTIFFRepresentationOfImageReps( CFArrayRef imageReps ) = CFDataRef
BitmapImageRepTIFFRepresentationOfImageRepsUsingCompression( CFArrayRef imageReps, NSTIFFCompression compression, float factor ) = CFDataRef
BitmapImageRepTIFFRepresentation( BitmapImageRepRef rep ) = CFDataRef
BitmapImageRepTIFFRepresentationUsingCompression( BitmapImageRepRef rep, NSTIFFCompression compression, float factor ) = CFDataRef
BitmapImageRepTIFFRepresentationOfImageRepsUsingType( CFArrayRef imageReps, NSBitmapImageFileType type, CFDictionaryRef properties ) = CFDataRef
BitmapImageRepRepresentationUsingType( BitmapImageRepRef rep, NSBitmapImageFileType type, CFDictionaryRef properties ) = CFDataRef
```

#### Compression

```
BitmapImageRepGetTIFFCompressionTypes( const NSTIFFCompression **list, NSInteger *count )
BitmapImageRepLocalizedStringForTIFFCompressionType( NSTIFFCompression compression ) = CFStringRef
BitmapImageRepCanBeCompressed( BitmapImageRepRef rep, NSTIFFCompression compression ) = Boolean
BitmapImageRepSetCompression( BitmapImageRepRef rep, NSTIFFCompression compression, float factor )
BitmapImageRepGetCompression( BitmapImageRepRef rep, NSTIFFCompression *compression, float *factor )
BitmapImageRepSetPropertyValue( BitmapImageRepRef rep, CFStringRef property, CFTypeRef value )
BitmapImageRepValueForProperty( BitmapImageRepRef rep, CFStringRef property ) = CFTypeRef
```

#### Load image incrementally

```
BitmapImageRepIncrementalLoadFromData( BitmapImageRepRef rep, CFDataRef dta, Boolean complete ) = NSInteger
```

#### Pixel values

```
BitmapImageRepSetColor( BitmapImageRepRef rep, ColorRef col, NSInteger x, NSInteger y )
BitmapImageRepColor( BitmapImageRepRef rep, NSInteger x, NSInteger y ) = ColorRef
BitmapImageRepSetPixel( BitmapImageRepRef rep, NSUInteger *p, NSInteger x, NSInteger y )
BitmapImageRepGetPixel( BitmapImageRepRef rep, NSUInteger *p, NSInteger x, NSInteger y )
```

#### CGImage

```
BitmapImageRepCGImage( BitmapImageRepRef rep ) = CGImageRef
```

#### Color spaces

```
BitmapImageRepByConvertingToColorSpace( BitmapImageRepRef rep, ColorSpaceRef cs, NSColorRenderingIntent intent ) = BitmapImageRepRef
BitmapImageRepByRetaggingWithColorSpace( BitmapImageRepRef rep, ColorSpaceRef cs ) = BitmapImageRepRef
BitmapImageRepColorSpace( BitmapImageRepRef rep ) = ColorSpaceRef
```

### Apple documentation

[NSBitmapImageRep](#)



Cocoa Box

statement

Syntax

```
cocoa box tag, title, rect, type, wndTag
```

Description

The **cocoa box** statement puts a new cocoa box in the current output cocoa window, or alters an existing cocoa box's characteristics.

Parameters

tag	A number to identify the box. A negative tag hides the box.
title	The box title.
rect	Origin and size of the box in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) CGRect value
type	The box type. NSBoxPrimary (default) NSBoxSecondary NSBoxSeparator NSBoxOldStyle NSBoxCustom For more information on type, see Apple's NSBox documentation.
wndTag	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

Functions

```
BoxWithTag( NSInteger tag ) = BoxRef
BoxExists( NSInteger tag ) = Boolean
```

```
Init
BoxInit( NSInteger tag, CGRect r ) = BoxRef// autoreleased
```

```
Configure
BoxBorderRect( NSInteger tag ) = CGRect
BoxType( NSInteger tag ) = NSBoxType
BoxSetType( NSInteger tag, NSBoxType type )
BoxBorderType( NSInteger tag ) = NSBorderType
BoxSetBorderType( NSInteger tag, NSInteger type )
BoxIsTransparent( NSInteger tag ) = Boolean
BoxSetTransparent( NSInteger tag, Boolean flag )
BoxTitle( NSInteger tag ) = CFStringRef
BoxSetTitle( NSInteger tag, CFStringRef title )
BoxTitleFont( NSInteger tag ) = CTNSFontRef
BoxSetTitleFont( NSInteger tag, CTNSFontRef font )
BoxSetTitleFontWithName( NSInteger tag, CFStringRef fontName, CGFloat size )
BoxTitlePosition( NSInteger tag ) = NSTitlePosition
BoxSetTitlePosition( NSInteger tag, NSInteger position )
BoxTitleRect( NSInteger tag ) = CGRect
```

```
Customizing
BoxBorderColor( NSInteger tag ) = ColorRef
BoxSetBorderColor( NSInteger tag, ColorRef col )
BoxBorderWidth( NSInteger tag ) = CGFloat
BoxSetBorderWidth( NSInteger tag, CGFloat width )
BoxCornerRadius( NSInteger tag ) = CGFloat
BoxSetCornerRadius( NSInteger tag, CGFloat radius )
BoxFillColor( NSInteger tag ) = ColorRef
BoxSetFillColor( NSInteger tag, ColorRef col )
```

```
Managing content
BoxContentViewMargins( NSInteger tag ) = CGSize
BoxSetContentViewMargins( NSInteger tag, CGSize size )
```

Sizing

```
BoxSetFrameFromContentFrame( NSInteger tag, CGRect contentFrame )  
BoxSizeToFit( NSInteger tag )
```

## See Also

[Control](#), [View](#)

## Apple documentation

[NSBox](#)



---

## Bundle

### Functions

```
BundleMain = BundleRef
BundleAllFrameworks = CFArrayRef
BundleAllBundles = CFArrayRef
```

```
Init
BundleWithURL( CFURLRef url ) = BundleRef
BundleWithIdentifier( CFStringRef identifier ) = BundleRef
```

```
Resource files
BundleURLForResource( BundleRef bundle, CFStringRef name, CFStringRef extension, CFStringRef subdirectory ) = CFURLRef
BundleURLsForResources( BundleRef bundle, CFStringRef extension, CFStringRef subdirectory ) = CFArrayRef// subdirectory can be NULL
```

```
Image resources
BundleURLForImageResource( BundleRef bundle, CFStringRef name ) = CFURLRef
BundleImageForResource( BundleRef bundle, CFStringRef name ) = ImageRef
```

```
Sound resources
BundleURLForSoundResource( BundleRef bundle, CFStringRef name ) = CFURLRef
```

```
Localized strings
BundleLocalizedString( BundleRef bundle, CFStringRef key, CFStringRef value, CFStringRef table ) = CFStringRef
```

```
Context help
BundleContextHelp( BundleRef bundle, CFStringRef key ) = CFAttributedStringRef
```

```
Standard bundle directories
BundleResourceURL( BundleRef bundle ) = CFURLRef
BundleExecutableURL( BundleRef bundle ) = CFURLRef
BundlePrivateFrameworksURL( BundleRef bundle ) = CFURLRef
BundleSharedFrameworksURL( BundleRef bundle ) = CFURLRef
BundleBuiltInPlugInsURL( BundleRef bundle ) = CFURLRef
BundleURLForAuxiliaryExecutable( BundleRef bundle, CFStringRef name ) = CFURLRef
BundleSharedSupportURL( BundleRef bundle ) = CFURLRef
BundleAppStoreReceiptURL( BundleRef bundle ) = CFURLRef
```

```
Bundle info
BundleURL( BundleRef bundle ) = CFURLRef
BundleIdentifier( BundleRef bundle ) = CFStringRef
BundleInfoDictionary( BundleRef bundle ) = CFDictionaryRef
BundleObjectForInfoDictionaryKey( BundleRef bundle, CFStringRef key ) = CFTypeRef
```

```
Localization info
BundleLocalizations( BundleRef bundle ) = CFArrayRef
BundlePreferredLocalizations( BundleRef bundle ) = CFArrayRef
BundleDevelopmentLocalization( BundleRef bundle ) = CFStringRef
BundleLocalizedInfoDictionary( BundleRef bundle ) = CFDictionaryRef
BundlePreferredLocalizationsFromArray( CFArrayRef localizationsArray ) = CFArrayRef
BundlePreferredLocalizationsFromArrayForPreferences( CFArrayRef localizationsArray, CFArrayRef preferencesArray ) = CFArrayRef
```

### Apple documentation

[NSBundle](#)



## Cocoa Button

statement

### Syntax

```
cocoa button tag, enabled, state, title, rect, type, style, wndTag
```

### Description

The **cocoa button** statement puts a new button in the current output cocoa window, or alters an existing button's characteristics.

### Parameters

<i>tag</i>	A number to identify the button. A negative tag hides the button.
<i>enabled</i>	Enables or disables the button.
<i>state</i>	The button's state. <a href="#">NSOnState</a> <a href="#">NSOffState</a> (default) <a href="#">NSMixedState</a>
<i>title</i>	The button's title. If a new button is given a title of "OK" or "Cancel", the button is automatically assigned a key equivalent of @“r” or @“e” respectively.
<i>rect</i>	Origin and size of the button in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>type</i>	The button type. <a href="#">NSMomentaryLightButton</a> (default) <a href="#">NSSwitchButton</a> <a href="#">NSRadioButton</a> <a href="#">NSMomentaryPushButton</a> <a href="#">NSPushOnPushOffButton</a> <a href="#">NSToggleButton</a> <a href="#">NSMomentaryChangeButton</a> <a href="#">NSOnOffButton</a> <a href="#">NSMomentaryPushInButton</a>
<i>style</i>	The style of the button. <a href="#">NSRoundedBezelStyle</a> (default) <a href="#">NSRegularSquareBezelStyle</a> <a href="#">NSThickSquareBezelStyle</a> <a href="#">NSThickerSquareBezelStyle</a> <a href="#">NSDisclosureBezelStyle</a> <a href="#">NSShadowlessSquareBezelStyle</a> <a href="#">NSCircularBezelStyle</a> <a href="#">NSTexturedSquareBezelStyle</a> <a href="#">NSHelpButtonBezelStyle</a> <a href="#">NSSmallSquareBezelStyle</a> <a href="#">NSTexturedRoundedBezelStyle</a> <a href="#">NSRoundRectBezelStyle</a> <a href="#">NSRecessedBezelStyle</a> <a href="#">NSRoundedDisclosureBezelStyle</a> <a href="#">NSInlineBezelStyle</a> <a href="#">NSSmallIconButtonBezelStyle</a>
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

[\\_btnClick](#)

### Functions

```
ButtonWithTag( NSInteger tag ) = ButtonRef  
ButtonExists( NSInteger tag ) = Boolean
```

Init

```
ButtonInit( NSInteger tag, CGRect r ) = ButtonRef// autoreleased
```

#### Configure

```
ButtonSetType( NSInteger tag, NSButtonType type )
ButtonGetPeriodicDelay( NSInteger tag, float *delay, float *interval )
ButtonSetPeriodicDelay( NSInteger tag, float delay, float interval )
ButtonAlternateTitle( NSInteger tag ) = CFStringRef
ButtonSetAlternateTitle( NSInteger tag, CFStringRef title )
ButtonAttributedTitle( NSInteger tag ) = CFAttributedStringRef
ButtonSetAttributedTitle( NSInteger tag, CFAttributedStringRef title )
ButtonAttributedAlternateTitle( NSInteger tag ) = CFAttributedStringRef
ButtonSetAttributedAlternateTitle( NSInteger tag, CFAttributedStringRef title )
ButtonTitle( NSInteger tag ) = CFStringRef
ButtonSetTitle( NSInteger tag, CFStringRef title )
ButtonSound( NSInteger tag ) = SoundRef
ButtonSetSound( NSInteger tag, SoundRef snd )
ButtonSetSoundNamed( NSInteger tag, CFStringRef name )
```

#### Configure images

```
ButtonImage( NSInteger tag ) = ImageRef
ButtonSetImage( NSInteger tag, ImageRef image )
ButtonSetImageNamed( NSInteger tag, CFStringRef imageName )
ButtonAlternateImage( NSInteger tag ) = ImageRef
ButtonSetAlternateImage( NSInteger tag, ImageRef image )
ButtonSetAlternateImageNamed( NSInteger tag, CFStringRef imageName )
ButtonImagePosition( NSInteger tag ) = NSCellImagePosition
ButtonSetImagePosition( NSInteger tag, NSCellImagePosition position )
ButtonIsBordered( NSInteger tag ) = Boolean
ButtonSetBordered( NSInteger tag, Boolean flag )
ButtonIsTransparent( NSInteger tag ) = Boolean
ButtonSetTransparent( NSInteger tag, Boolean flag )
ButtonBezelStyle( NSInteger tag ) = NSBezelStyle
ButtonSetBezelStyle( NSInteger tag, NSBezelStyle style )
ButtonBezelColor( NSInteger tag ) = ColorRefmacOS 10.12.1+
ButtonSetBezelColor( NSInteger tag, ColorRef col )macOS 10.12.1+
ButtonShowsBorderOnlyWhileMouseInside( NSInteger tag ) = Boolean
ButtonSetShowsBorderOnlyWhileMouseInside( NSInteger tag, Boolean flag )
ButtonImageHugsTitle( NSInteger tag ) = BooleanmacOS 10.12+
ButtonSetImageHugsTitle( NSInteger tag, Boolean flag )macOS 10.12+
ButtonImageScaling( NSInteger tag ) = NSImageScaling
ButtonSetImageScaling( NSInteger tag, NSImageScaling scaling )
```

#### State

```
ButtonAllowsMixedState( NSInteger tag ) = Boolean
ButtonSetAllowsMixedState( NSInteger tag, Boolean flag )
ButtonState( NSInteger tag ) = NSInteger
ButtonSetState( NSInteger tag, NSCellStateValue state )
ButtonSetNextState( NSInteger tag )
ButtonHighlight( NSInteger tag, Boolean flag )
```

#### Key equivalents

```
ButtonKeyEquivalent( NSInteger tag ) = CFStringRef
ButtonSetKeyEquivalent( NSInteger tag, CFStringRef key )
ButtonKeyEquivalentModifierMask( NSInteger tag ) = NSUInteger
ButtonSetKeyEquivalentModifierMask( NSInteger tag, NSEventModifierFlags mask )
ButtonKeyEquivalentFont( NSInteger tag ) = CTNSFontRef
ButtonSetKeyEquivalentFont( NSInteger tag, CTNSFontRef font )
```

#### Convenience

```
ButtonSetBackgroundColor( NSInteger tag, ColorRef col )
ButtonImageDimsWhenDisabled( NSInteger tag ) = Boolean
ButtonSetImageDimsWhenDisabled( NSInteger tag, Boolean flag )
ButtonIsOpaque( NSInteger tag ) = Boolean
ButtonSetHighlightsBy( NSInteger tag, NSCellStyleMask mask )
ButtonPerformClick( NSInteger tag )
```

#### - custom -

```
ButtonClose( NSInteger tag )
ButtonSetTitleColor( NSInteger tag, ColorRef col )
```













## See Also

[Control](#), [View](#)

## Apple documentation

[NSButton](#)



Cocoa Button Varieties				Reg		Small		Mini	
Kind		Type	Style	w	h	w	h	w	h
	Push		NSRoundedBezelStyle	-	32	-	28	-	16
	Textured Rounded		NSTexturedRoundedBezelStyle	-	25	-	19	-	16
	Gradient		NSSmallSquareBezelStyle	-	-	-	-	-	-
	Check Box	NSSwitchButton		-	18	-	18	-	18
	Radio	NSRadioButton		-	18	-	18	-	18
	Round Rect		NSRoundRectBezelStyle	-	19	-	17	-	17
	Recessed	NSPushOnPushOffButton	NSRecessedBezelStyle	-	19	-	17	-	17
	Inline		NSInlineBezelStyle	-	17	-	-	-	-
	Image		NSShadowlessSquareBezelStyle	-	-	-	-	-	-
	Disclosure	NSOnOffButton	NSRoundedDisclosureBezelStyle	29	26	25	23	17	16
	Disclosure Triangle		NSDisclosureBezelStyle	13	13	-	-	-	-
	Help		NSHelpButtonBezelStyle	25	25	-	-	-	-



---

## ByteCountFormatter

### Functions

Create

```
ByteCountFormatterInit = ByteCountFormatterRef// autoreleased
```

String from byte count

```
ByteCountFormatterStringFromByteCount( SInt64 byteCount, NSByteCountFormatterCountStyle style ) = CFStringRef
```

Formatting styles

```
ByteCountFormatterFormattingContext( ByteCountFormatterRef ref ) = NSFormattingContext// macOS 10.10+
```

```
ByteCountFormatterSetFormattingContext( ByteCountFormatterRef ref, NSFormattingContext ctx )// macOS 10.10+
```

```
ByteCountFormatterCountStyle( ByteCountFormatterRef ref ) = NSByteCountFormatterCountStyle
```

```
ByteCountFormatterSetCountStyle( ByteCountFormatterRef ref, NSByteCountFormatterCountStyle style )
```

```
ByteCountFormatterAllowsNonnumericFormatting( ByteCountFormatterRef ref ) = Boolean
```

```
ByteCountFormatterSetAllowsNonnumericFormatting( ByteCountFormatterRef ref, Boolean flag )
```

```
ByteCountFormatterIncludesActualByteCount( ByteCountFormatterRef ref ) = Boolean
```

```
ByteCountFormatterSetIncludesActualByteCount( ByteCountFormatterRef ref, Boolean flag )
```

```
ByteCountFormatterIsAdaptive( ByteCountFormatterRef ref ) = Boolean
```

```
ByteCountFormatterSetAdaptive( ByteCountFormatterRef ref, Boolean flag )
```

```
ByteCountFormatterAllowedUnits( ByteCountFormatterRef ref ) = NSByteCountFormatterUnits
```

```
ByteCountFormatterSetAllowedUnits( ByteCountFormatterRef ref, NSByteCountFormatterUnits units )
```

```
ByteCountFormatterIncludesCount( ByteCountFormatterRef ref ) = Boolean
```

```
ByteCountFormatterSetIncludesCount( ByteCountFormatterRef ref, Boolean flag )
```

```
ByteCountFormatterZeroPadsFractionDigits( ByteCountFormatterRef ref ) = Boolean
```

```
ByteCountFormatterSetZeroPadsFractionDigits( ByteCountFormatterRef ref, Boolean flag )
```

### Apple documentation

[NSByteCountFormatter](#)



---

## CAAnimation

### Functions

Create

```
CAAnimationInit = CAAnimationRef// autoreleased
```

Attributes

```
CAAnimationIsRemovedOnCompletion( CAAnimationRef ref ) = Boolean  
CAAnimationSetRemovedOnCompletion( CAAnimationRef ref, Boolean value )  
CAAnimationTimingFunction( CAAnimationRef ref ) = CAMediaTimingFunctionRef  
CAAnimationSetTimingFunction( CAAnimationRef ref, CAMediaTimingFunctionRef value )
```

Default values

```
CAAnimationDefaultValueForKey = CFTypeRef  
CAAnimationSetDefaultValueForKey( CFTypeRef value )
```

Archiving

```
CAAnimationShouldArchiveValueForKey( CAAnimationRef ref, CFStringRef key ) = Boolean
```

### Apple documentation

[CAAnimation](#)



---

## CABasicAnimation

### Functions

Create

```
CABasicAnimationWithKeyPath( CFStringRef path ) = CABasicAnimationRef
```

```
// Interpolation values
```

```
CABasicAnimationFromValue( CABasicAnimationRef ref ) = CFTypeRef
```

```
CABasicAnimationSetFromValue( CABasicAnimationRef ref, CFTypeRef value )
```

```
CABasicAnimationToValue( CABasicAnimationRef ref ) = CFTypeRef
```

```
CABasicAnimationSetToValue( CABasicAnimationRef ref, CFTypeRef value )
```

```
CABasicAnimationByValue( CABasicAnimationRef ref ) = CFTypeRef
```

```
CABasicAnimationSetByValue( CABasicAnimationRef ref, CFTypeRef value )
```

### Apple documentation

[CABasicAnimation](#)



---

## CAGradientLayer

### Functions

Init

```
CAGradientLayerInit = CAGradientLayerRef// autoreleased
```

Style

```
CAGradientLayerColors( CAGradientLayerRef layer ) = CFArrayRef// macOS 10.8+      // array of ColorRefs
CAGradientLayerSetColors( CAGradientLayerRef layer, CFArrayRef cols )// macOS 10.8+      // array of ColorRefs
CAGradientLayerLocations( CAGradientLayerRef layer ) = CFArrayRef// array of CFNumberRefs (values between 0 and 1)
CAGradientLayerSetLocations( CAGradientLayerRef layer, CFArrayRef locations )// array of CFNumberRefs - can be NULL
CAGradientEndPoint( CAGradientLayerRef layer ) = CGPoint
CAGradientLayerSetEndPoint( CAGradientLayerRef layer, CGPoint pt )
CAGradientStartPoint( CAGradientLayerRef layer ) = CGPoint
CAGradientLayerSetStartPoint( CAGradientLayerRef layer, CGPoint pt )
CAGradientType( CAGradientLayerRef layer ) = CFStringRef
CAGradientLayerSetType( CAGradientLayerRef layer, CFStringRef type )
```

Convenience

```
CAGradientLayerWithStartEndColors( CGRect r, ColorRef startColor, CGPoint startPt, ColorRef endColor, CGPoint
endPt ) = CAGradientLayerRef// macOS 10.8+
```

### Apple documentation

[CAGradientLayer](#)



---

## CAKeyframeAnimation

### Functions

Create

```
CAKeyframeAnimationWithKeyPath( CFStringRef path ) = CAKeyframeAnimationRef
```

Values

```
CAKeyframeAnimationValues( CAKeyframeAnimationRef ref ) = CFArrayRef  
CAKeyframeAnimationSetValues( CAKeyframeAnimationRef ref, CFArrayRef value )  
CAKeyframeAnimationPath( CAKeyframeAnimationRef ref ) = CGPathRef  
CAKeyframeAnimationSetPath( CAKeyframeAnimationRef ref, CGPathRef value )
```

Timing

```
CAKeyframeAnimationKeyTimes( CAKeyframeAnimationRef ref ) = CFArrayRef  
CAKeyframeAnimationSetKeyTimes( CAKeyframeAnimationRef ref, CFArrayRef value )  
CAKeyframeAnimationTimingFunctions( CAKeyframeAnimationRef ref ) = CFArrayRef  
CAKeyframeAnimationSetTimingFunctions( CAKeyframeAnimationRef ref, CFArrayRef value )  
CAKeyframeAnimationCalculationMode( CAKeyframeAnimationRef ref ) = CFStringRef  
CAKeyframeAnimationSetCalculationMode( CAKeyframeAnimationRef ref, CFStringRef value )
```

Rotation mode

```
CAKeyframeAnimationRotationMode( CAKeyframeAnimationRef ref ) = CFStringRef  
CAKeyframeAnimationSetRotationMode( CAKeyframeAnimationRef ref, CFStringRef value )
```

Cubic mode

```
CAKeyframeAnimationTensionValues( CAKeyframeAnimationRef ref ) = CFArrayRef  
CAKeyframeAnimationSetTensionValues( CAKeyframeAnimationRef ref, CFArrayRef value )  
CAKeyframeAnimationContinuityValues( CAKeyframeAnimationRef ref ) = CFArrayRef  
CAKeyframeAnimationSetContinuityValues( CAKeyframeAnimationRef ref, CFArrayRef value )  
CAKeyframeAnimationBiasValues( CAKeyframeAnimationRef ref ) = CFArrayRef  
CAKeyframeAnimationSetBiasValues( CAKeyframeAnimationRef ref, CFArrayRef value )
```

### Apple documentation

[CAKeyframeAnimation](#)



## CALayer

### Functions

Create

```
CALayerInit = CALayerRef// autoreleased
```

Content

```
CALayerContents( CALayerRef layer ) = ptr  
CALayerSetContents( CALayerRef layer, ptr contents )  
CALayerContentsRect( CALayerRef layer ) = CGRect  
CALayerSetContentsRect( CALayerRef layer, CGRect r )  
CALayerContentsCenter( CALayerRef layer ) = CGRect  
CALayerSetContentsCenter( CALayerRef layer, CGRect r )
```

Appearance

```
CALayerContentsGravity( CALayerRef layer ) = CFStringRef  
CALayerSetContentsGravity( CALayerRef layer, CFStringRef gravity )  
CALayerOpacity( CALayerRef layer ) = float  
CALayerSetOpacity( CALayerRef layer, float opacity )  
CALayerIsHidden( CALayerRef layer ) = Boolean  
CALayerSetHidden( CALayerRef layer, Boolean flag )  
CALayerMasksToBounds( CALayerRef layer ) = Boolean  
CALayerSetMasksToBounds( CALayerRef layer, Boolean flag )  
CALayerMask( CALayerRef layer ) = CALayerRef  
CALayerSetMask( CALayerRef layer, CALayerRef mask )  
CALayerIsDoubleSided( CALayerRef layer ) = Boolean  
CALayerSetDoubleSided( CALayerRef layer, Boolean flag )  
CALayerCornerRadius( CALayerRef layer ) = CGFloat  
CALayerSetCornerRadius( CALayerRef layer, CGFloat radius )  
CALayerMaskedCorners( CALayerRef layer ) = CACornerMask// macOS 10.13+  
CALayerSetMaskedCorners( CALayerRef layer, CACornerMask mask )// macOS 10.13+  
CALayerBorderWidth( CALayerRef layer ) = CGFloat  
CALayerSetBorderWidth( CALayerRef layer, CGFloat width )  
CALayerBorderColor( CALayerRef layer ) = ColorRef// macOS 10.8+  
CALayerSetBorderColor( CALayerRef layer, ColorRef col )// macOS 10.8+  
CALayerBackgroundColor( CALayerRef layer ) = ColorRef// macOS 10.8+  
CALayerSetBackgroundColor( CALayerRef layer, ColorRef col )// macOS 10.8+  
CALayerShadowOpacity( CALayerRef layer ) = float  
CALayerSetShadowOpacity( CALayerRef layer, float opacity )  
CALayerShadowRadius( CALayerRef layer ) = CGFloat  
CALayerSetShadowRadius( CALayerRef layer, CGFloat radius )  
CALayerShadowOffset( CALayerRef layer ) = CGSize  
CALayerSetShadowOffset( CALayerRef layer, CGSize offset )  
CALayerShadowColor( CALayerRef layer ) = ColorRef// macOS 10.8+  
CALayerSetShadowColor( CALayerRef layer, ColorRef col )// macOS 10.8+  
CALayerShadowPath( CALayerRef layer ) = CGPathRef  
CALayerSetShadowPath( CALayerRef layer, CGPathRef path )  
CALayerStyle( CALayerRef layer ) = CFDictionaryRef  
CALayerSetStyle( CALayerRef layer, CFDictionaryRef style )
```

Rendering behavior

```
CALayerIsOpaque( CALayerRef layer ) = Boolean  
CALayerSetOpaque( CALayerRef layer, Boolean flag )  
CALayerContentsAreFlipped( CALayerRef layer ) = Boolean
```

Geometry

```
CALayerFrame( CALayerRef layer ) = CGRect  
CALayerSetFrame( CALayerRef layer, CGRect frame )  
CALayerBounds( CALayerRef layer ) = CGRect  
CALayerSetBounds( CALayerRef layer, CGRect bounds )  
CALayerPosition( CALayerRef layer ) = CGPoint  
CALayerSetPosition( CALayerRef layer, CGPoint pt )  
CALayerZPosition( CALayerRef layer ) = CGFloat  
CALayerSetZPosition( CALayerRef layer, CGFloat position )  
CALayerAnchorPointZ( CALayerRef layer ) = CGFloat  
CALayerSetAnchorPointZ( CALayerRef layer, CGFloat position )  
CALayerAnchorPoint( CALayerRef layer ) = CGPoint  
CALayerSetAnchorPoint( CALayerRef layer, CGPoint pt )  
CALayerContentsScale( CALayerRef layer ) = CGFloat  
CALayerSetContentsScale( CALayerRef layer, CGFloat scale )
```

## Transform

```
CALayerTransform( CALayerRef layer ) = CATransform3D
CALayerSetTransform( CALayerRef layer, CATransform3D t )
CALayerSublayerTransform( CALayerRef layer ) = CATransform3D
CALayerSetSublayerTransform( CALayerRef layer, CATransform3D t )
CALayerAffineTransform( CALayerRef layer ) = CGAffineTransform
CALayerSetAffineTransform( CALayerRef layer, CGAffineTransform t )
```

## Hierarchy

```
CALayerSublayers( CALayerRef layer ) = CFArrayRef
CALayerSetSublayers( CALayerRef layer, CFArrayRef sublayers )
CALayerSuperlayer( CALayerRef layer ) = CALayerRef
CALayerAddSublayer( CALayerRef layer, CALayerRef sublayer )
CALayerRemoveFromSuperlayer( CALayerRef layer )
CALayerInsertSublayer( CALayerRef layer, CALayerRef otherLayer, unsigned long index )
CALayerInsertSublayerBelow( CALayerRef layer, CALayerRef insertLayer, CALayerRef sibling )
CALayerInsertSublayerAbove( CALayerRef layer, CALayerRef insertLayer, CALayerRef sibling )
CALayerReplaceSublayer( CALayerRef layer, CALayerRef subLayer, CALayerRef otherLayer )
```

## Display

```
CALayerSetNeedsDisplay( CALayerRef layer )
```

## Animations

```
CALayerAddAnimation( CALayerRef layer, CAAAnimationRef anim, CFStringRef key )
CALayerAnimation( CALayerRef layer, CFStringRef key ) = CAAAnimationRef
CALayerRemoveAllAnimations( CALayerRef layer )
CALayerRemoveAnimation( CALayerRef layer, CFStringRef key )
CALayerAnimationKeys( CALayerRef layer ) = CFArrayRef
```

## Resizing and layout

```
CALayerAutoresizingMask( CALayerRef layer ) = NSUInteger
CALayerSetAutoresizingMask( CALayerRef layer, NSUInteger mask )
```

## Hit testing

```
CALayerHitTest( CALayerRef layer, CGPoint pt ) = CALayerRef
CALayerContainsPoint( CALayerRef layer, CGPoint pt ) = Boolean
```

## Scrolling

```
CALayerVisibleRect( CALayerRef layer ) = CGRect
CALayerScrollPoint( CALayerRef layer, CGPoint pt )
CALayerScrollRectToVisible( CALayerRef layer, CGRect r )
```

## Identifying

```
CALayerName( CALayerRef layer ) = CFStringRef
CALayerSetName( CALayerRef layer, CFStringRef name )
```

## Apple documentation

[CALayer](#)





## Calendar

### Functions

Create

```
CalendarWithIdentifier( CFStringRef identifier ) = CFCalendarRef
```

User's calendar

```
CalendarCurrent = CFCalendarRef
```

```
CalendarAutoupdatingCurrent = CFCalendarRef
```

Components

```
CalendarDateMatches( CFCalendarRef cal, CFDateRef dt, DateComponentsRef comp ) = Boolean// macOS 10.9+
```

```
CalendarComponentFromDate( CFCalendarRef cal, NSUInteger unit, CFDateRef dt ) = NSInteger// macOS 10.9+
```

```
CalendarComponentsFromDate( CFCalendarRef cal, NSUInteger unitFlags, CFDateRef dt ) = DateComponentsRef
```

```
CalendarComponentsFromDateToDate( CFCalendarRef cal, NSUInteger unitFlags, CFDateRef fromDate, CFDateRef toDate, NSUInteger options ) = DateComponentsRef
```

```
CalendarComponentsFromDateComponentsToDateComponents( CFCalendarRef cal, NSUInteger unitFlags, DateComponentsRef fromComp, DateComponentsRef toComp, NSUInteger options ) = DateComponentsRef// macOS 10.9+
```

```
CalendarComponentsInTimeZoneFromDate( CFCalendarRef cal, CFTimeZoneRef zone, CFDateRef dt ) = DateComponentsRef
```

```
CalendarGetEraYearMonthDayFromDate( CFCalendarRef cal, NSInteger *era, NSInteger *year, NSInteger month, NSInteger day, CFDateRef dt )
```

```
CalendarGetHourMinuteSecondNanosecondFromDate( CFCalendarRef cal, NSInteger *hour, NSInteger *min, NSInteger second, NSInteger nanosecond, CFDateRef dt )
```

Info

```
CalendarIdentifier( CFCalendarRef cal ) = CFStringRef
```

```
CalendarFirstWeekday( CFCalendarRef cal ) = NSUInteger
```

```
CalendarLocale( CFCalendarRef cal ) = CFLocaleRef
```

```
CalendarTimeZone( CFCalendarRef cal ) = CFTimeZoneRef
```

```
CalendarMaximumRangeOfUnit( CFCalendarRef cal, NSCalendarUnit unit ) = CFRange
```

```
CalendarMinimumRangeOfUnit( CFCalendarRef cal, NSCalendarUnit unit ) = CFRange
```

```
CalendarMinimumDaysInFirstWeek( CFCalendarRef cal ) = NSUInteger
```

```
CalendarOrdinalityOfUnitInUnit( CFCalendarRef cal, NSUInteger ofUnit, NSUInteger inUnit, CFDateRef dt ) = NSUInteger
```

```
CalendarRangeOfUnitInUnit( CFCalendarRef cal, NSUInteger ofUnit, NSUInteger inUnit, CFDateRef dt ) = CFRange
```

```
CalendarRangeOfUnit( CFCalendarRef cal, NSUInteger unit, CFDateRef *startDate, CFTimeInterval *interval, CFDateRef forDate ) = Boolean
```

```
CalendarRangeOfWeekend( CFCalendarRef cal, CFDateRef startDate, CFTimeInterval interval, CFDateRef containingDate ) = CFRange
```

Scanning dates

```
CalendarStartOfDayForDate( CFCalendarRef cal, CFDateRef dt ) = CFDateRef// macOS 10.9+
```

Callback example:

```
local fn MyEnumerateDates( cal as CFCalendarRef, dt as CFDateRef, exactMatch as Boolean, userData as ptr ) as Boolean
```

```
end fn = _false // return _true to stop enumeration
```

```
CalendarEnumerateDates( CFCalendarRef cal, CFDateRef startDate, DateComponentsRef matchingComponents, NSUInteger options, ptr callback, ptr userData )// macOS 10.9+
```

```
CalendarNextDateAfterDateMatchingComponents( CFCalendarRef cal, CFDateRef dt, DateComponentsRef comps, NSUInteger options ) = CFDateRef// macOS 10.9+
```

```
CalendarNextDateAfterDateMatchingHourMinuteSeconds( CFCalendarRef cal, CFDateRef dt, NSInteger hour, NSInteger minute, NSInteger second, NSUInteger options ) = CFDateRef// macOS 10.9+
```

```
CalendarNextDateAfterDateMatchingUnit( CFCalendarRef cal, CFDateRef dt, NSUInteger unit, NSUInteger options ) = CFDateRef// macOS 10.9+
```

Calculating dates

```
CalendarDateFromComponents( CFCalendarRef cal, DateComponentsRef comps ) = CFDateRef
```

```
CalendarDateByAddingComponents( CFCalendarRef cal, DateComponentsRef comps, CFDateRef dt, NSUInteger options ) = CFDateRef
```

```
CalendarDateByAddingUnit( CFCalendarRef cal, NSUInteger unit, NSInteger value, CFDateRef dt, NSUInteger options ) = CFDateRef// macOS 10.9+
```

```
CalendarDateBySettingHourMinuteSecond( CFCalendarRef cal, NSInteger hour, NSInteger minute, NSInteger second, CFDateRef dt, NSUInteger options ) = CFDateRef// macOS 10.9+
```

```
CalendarDateBySettingUnit( CFCalendarRef cal, NSUInteger unit, NSInteger value, CFDateRef dt, NSUInteger options ) = CFDateRef// macOS 10.9+
```

```
CalendarDateWithEra( CFCalendarRef cal, NSInteger era, NSInteger year, NSInteger month, NSInteger day, NSInteger hour, NSInteger minute, NSInteger second, NSInteger nanosecond ) = CFDateRef// macOS 10.9+
```

```
CalendarDateWithEraYearForWeekOfYear( CFCalendarRef cal, NSInteger era, NSInteger year, NSInteger week, NSInteger weekday, NSInteger hour, NSInteger minute, NSInteger second, NSInteger nanosecond ) = CFDateRef// macOS 10.9+
```

```
CalendarNextWeekendStartDate( CFCalendarRef cal, CFDateRef *dt, CFTimeInterval *interval, NSUInteger options, CFDateRef afterDate ) = Boolean// macOS 10.9+
```

Comparing dates

```
CalendarCompareDateToDate( CFCalendarRef cal, CFDateRef dt1, CFDateRef dt2, NSCalendarUnit unit ) =
NSComparisonResult
CalendarIsDateEqualToDate( CFCalendarRef cal, CFDateRef dt1, CFDateRef dt2, NSCalendarUnit unit ) = Boolean
CalendarIsDateInSameDayAsDate( CFCalendarRef cal, CFDateRef dt1, CFDateRef dt2 ) = Boolean
CalendarIsDateInToday( CFCalendarRef cal, CFDateRef dt ) = Boolean
CalendarIsDateInTomorrow( CFCalendarRef cal, CFDateRef dt ) = Boolean
CalendarIsDateInWeekend( CFCalendarRef cal, CFDateRef dt ) = Boolean
CalendarIsDateInYesterday( CFCalendarRef cal, CFDateRef dt ) = Boolean
```

AM and PM symbols

```
CalendarAMSymbol( CFCalendarRef cal ) = CFStringRef
CalendarPMSymbol( CFCalendarRef cal ) = CFStringRef
```

Weekday symbols

```
CalendarWeekdaySymbols( CFCalendarRef cal ) = CFArrayRef
CalendarShortWeekdaySymbols( CFCalendarRef cal ) = CFArrayRef
CalendarVeryShortWeekdaySymbols( CFCalendarRef cal ) = CFArrayRef
CalendarStandaloneWeekdaySymbols( CFCalendarRef cal ) = CFArrayRef
CalendarShortStandaloneWeekdaySymbols( CFCalendarRef cal ) = CFArrayRef
CalendarVeryShortStandaloneWeekdaySymbols( CFCalendarRef cal ) = CFArrayRef
```

Month symbols

```
CalendarMonthSymbols( CFCalendarRef cal ) = CFArrayRef
CalendarShortMonthSymbols( CFCalendarRef cal ) = CFArrayRef
CalendarVeryShortMonthSymbols( CFCalendarRef cal ) = CFArrayRef
CalendarStandaloneMonthSymbols( CFCalendarRef cal ) = CFArrayRef
CalendarShortStandaloneMonthSymbols( CFCalendarRef cal ) = CFArrayRef
CalendarVeryShortStandaloneMonthSymbols( CFCalendarRef cal ) = CFArrayRef
```

Quarter symbols

```
CalendarQuarterSymbols( CFCalendarRef cal ) = CFArrayRef
CalendarShortQuarterSymbols( CFCalendarRef cal ) = CFArrayRef
CalendarStandaloneQuarterSymbols( CFCalendarRef cal ) = CFArrayRef
CalendarShortStandaloneQuarterSymbols( CFCalendarRef cal ) = CFArrayRef
```

Era symbols

```
CalendarEraSymbols( CFCalendarRef cal ) = CFArrayRef
CalendarLongEraSymbols( CFCalendarRef cal ) = CFArrayRef
```

## Apple documentation

[NSCalendar](#)



---

## CAMediaTiming

### Functions

Animation start time

```
CAMediaTimingBeginTime( CAMediaTimingRef ref ) = CFTimeInterval  
CAMediaTimingSetBeginTime( CAMediaTimingRef ref, CFTimeInterval ti )  
CAMediaTimingTimeOffset( CAMediaTimingRef ref ) = CFTimeInterval  
CAMediaTimingSetTimeOffset( CAMediaTimingRef ref, CFTimeInterval ti )
```

Repeating animations

```
CAMediaTimingRepeatCount( CAMediaTimingRef ref ) = float  
CAMediaTimingSetRepeatCount( CAMediaTimingRef ref, float count )  
CAMediaTimingRepeatDuration( CAMediaTimingRef ref ) = CFTimeInterval  
CAMediaTimingSetRepeatDuration( CAMediaTimingRef ref, CFTimeInterval ti )
```

Duration and speed

```
CAMediaTimingDuration( CAMediaTimingRef ref ) = CFTimeInterval  
CAMediaTimingSetDuration( CAMediaTimingRef ref, CFTimeInterval ti )  
CAMediaTimingSpeed( CAMediaTimingRef ref ) = float  
CAMediaTimingSetSpeed( CAMediaTimingRef ref, float speed )
```

Playback modes

```
CAMediaTimingAutoreverses( CAMediaTimingRef ref ) = Boolean  
CAMediaTimingSetAutoreverses( CAMediaTimingRef ref, Boolean flag )  
CAMediaTimingFillMode( CAMediaTimingRef ref ) = CFStringRef  
CAMediaTimingSetFillMode( CAMediaTimingRef ref, CFStringRef mode )
```

### Apple documentation

[CAMediaTiming](#)



---

## CAMediaTimingFunction

### Functions

Create

```
CAMediaTimingFunctionWithName( CFStringRef name ) = CAMediaTimingFunctionRef
```

```
CAMediaTimingFunctionWithControlPoints( float c1x, float c1y, float c2x, float c2y ) = CAMediaTimingFunctionRef
```

Control points

```
CAMediaTimingFunctionGetControlPointAtIndex( CAMediaTimingFunctionRef ref, float *pt )
```

### Apple documentation

[CAMediaTimingFunction](#)



---

## CAPROPERTYANIMATION

### Functions

Create

```
CAPROPERTYANIMATIONWITHKEYPATH( CFStringRef path ) = CAPROPERTYANIMATIONREF
```

Key path

```
CAPROPERTYANIMATIONKEYPATH( CAPROPERTYANIMATIONREF ref ) = CFStringRef
```

```
CAPROPERTYANIMATIONSETKEYPATH( CAPROPERTYANIMATIONREF ref, CFStringRef value )
```

Property value calculation behavior

```
CAPROPERTYANIMATIONISCUMULATIVE( CAPROPERTYANIMATIONREF ref ) = Boolean
```

```
CAPROPERTYANIMATIONSETCUMULATIVE( CAPROPERTYANIMATIONREF ref, Boolean value )
```

```
CAPROPERTYANIMATIONISADDITIVE( CAPROPERTYANIMATIONREF ref ) = Boolean
```

```
CAPROPERTYANIMATIONSETADDITIVE( CAPROPERTYANIMATIONREF ref, Boolean value )
```

```
CAPROPERTYANIMATIONVALUEFUNCTION( CAPROPERTYANIMATIONREF ref ) = CAValueFunctionRef
```

```
CAPROPERTYANIMATIONSETVALUEFUNCTION( CAPROPERTYANIMATIONREF ref, CAValueFunctionRef value )
```

### Apple documentation

[CAPROPERTYANIMATION](#)



---

## CAReplicatorLayer

### Functions

Init  
`CAReplicatorLayerInit = CAReplicatorLayerRef// autoreleased`

Display  
`CAReplicatorLayerInstanceCount( CAReplicatorLayerRef layer ) = NSInteger`  
`CAReplicatorLayerSetInstanceCount( CAReplicatorLayerRef layer, NSInteger count )`  
`CAReplicatorLayerInstanceDelay( CAReplicatorLayerRef layer ) = CFTimeInterval`  
`CAReplicatorLayerSetInstanceDelay( CAReplicatorLayerRef layer, CFTimeInterval delay )`  
`CAReplicatorLayerInstanceTransform( CAReplicatorLayerRef layer ) = CATransform3D`  
`CAReplicatorLayerSetInstanceTransform( CAReplicatorLayerRef layer, CATransform3D t )`

Geometry  
`CAReplicatorLayerPreservesDepth( CAReplicatorLayerRef layer ) = Boolean`  
`CAReplicatorLayerSetPreservesDepth( CAReplicatorLayerRef layer, Boolean flag )`

Color  
`CAReplicatorLayerInstanceColor( CAReplicatorLayerRef layer ) = ColorRef// macOS 10.8+`  
`CAReplicatorLayerSetInstanceColor( CAReplicatorLayerRef layer, ColorRef col )// macOS 10.8+`  
`CAReplicatorLayerInstanceRedOffset( CAReplicatorLayerRef layer ) = float`  
`CAReplicatorLayerSetInstanceRedOffset( CAReplicatorLayerRef layer, float offset )`  
`CAReplicatorLayerInstanceGreenOffset( CAReplicatorLayerRef layer ) = float`  
`CAReplicatorLayerSetInstanceGreenOffset( CAReplicatorLayerRef layer, float offset )`  
`CAReplicatorLayerInstanceBlueOffset( CAReplicatorLayerRef layer ) = float`  
`CAReplicatorLayerSetInstanceBlueOffset( CAReplicatorLayerRef layer, float offset )`  
`CAReplicatorLayerInstanceAlphaOffset( CAReplicatorLayerRef layer ) = float`  
`CAReplicatorLayerSetInstanceAlphaOffset( CAReplicatorLayerRef layer, float offset )`

### Apple documentation

[CAReplicatorLayer](#)



---

## CAScrollLayer

### Functions

Init

```
CAScrollLayerInit = CAScrollLayerRef// autoreleased
```

Constraints

```
CAScrollLayerScrollMode( CAScrollLayerRef layer ) = CFStringRef
```

```
CAScrollLayerSetScrollMode( CAScrollLayerRef layer, CFStringRef scrollMode )
```

Scrolling

```
CAScrollLayerScrollToPoint( CAScrollLayerRef layer, CGPoint pt )
```

```
CAScrollLayerScrollToRect( CAScrollLayerRef layer, CGRect r )
```

### Apple documentation

[CAScrollLayer](#)



---

## CAShapeLayer

### Functions

Init

```
CAShapeLayerInit = CAShapeLayerRef// autoreleased
```

Path

```
//CAShapeLayerPath( CAShapeLayerRef layer ) = BezierPathRef  
CAShapeLayerSetPath( CAShapeLayerRef layer, BezierPathRef path )
```

Properties

```
CAShapeLayerFillColor( CAShapeLayerRef layer ) = ColorRef// macOS 10.8+  
CAShapeLayerSetFillColor( CAShapeLayerRef layer, ColorRef fillColor )// macOS 10.8+  
CAShapeLayerFillRule( CAShapeLayerRef layer ) = CFStringRef  
CAShapeLayerSetFillRule( CAShapeLayerRef layer, CFStringRef fillRule )  
CAShapeLayerLineCap( CAShapeLayerRef layer ) = CFStringRef  
CAShapeLayerSetLineCap( CAShapeLayerRef layer, CFStringRef lineCap )  
CAShapeLayerLineDashPattern( CAShapeLayerRef layer ) = CFArrayRef  
CAShapeLayerSetLineDashPattern( CAShapeLayerRef layer, CFArrayRef lineDashPattern )  
CAShapeLayerLineDashPhase( CAShapeLayerRef layer ) = CGFloat  
CAShapeLayerSetLineDashPhase( CAShapeLayerRef layer, CGFloat lineDashPhase )  
CAShapeLayerLineJoin( CAShapeLayerRef layer ) = CFStringRef  
CAShapeLayerSetLineJoin( CAShapeLayerRef layer, CFStringRef lineJoin )  
CAShapeLayerLineWidth( CAShapeLayerRef layer ) = CGFloat  
CAShapeLayerSetLineWidth( CAShapeLayerRef layer, CGFloat lineWidth )  
CAShapeLayerMiterLimit( CAShapeLayerRef layer ) = CGFloat  
CAShapeLayerSetMiterLimit( CAShapeLayerRef layer, CGFloat miterLimit )  
CAShapeLayerStrokeColor( CAShapeLayerRef layer ) = ColorRef// macOS 10.8+  
CAShapeLayerSetStrokeColor( CAShapeLayerRef layer, ColorRef strokeColor )// macOS 10.8+  
CAShapeLayerStrokeStart( CAShapeLayerRef layer ) = CGFloat  
CAShapeLayerSetStrokeStart( CAShapeLayerRef layer, CGFloat strokeStart )  
CAShapeLayerStrokeEnd( CAShapeLayerRef layer ) = CGFloat  
CAShapeLayerSetStrokeEnd( CAShapeLayerRef layer, CGFloat strokeEnd )
```

### Apple documentation

[CAShapeLayer](#)





---

## CASpringAnimation

### Functions

Create

```
CASpringAnimationWithKeyPath( CFStringRef path ) = CASpringAnimationRef
```

Attributes

```
CASpringAnimationDamping( CASpringAnimationRef ref ) = CGFloat  
CASpringAnimationSetDamping( CASpringAnimationRef ref, CGFloat value )  
CASpringAnimationInitialVelocity( CASpringAnimationRef ref ) = CGFloat  
CASpringAnimationSetInitialVelocity( CASpringAnimationRef ref, CGFloat value )  
CASpringAnimationMass( CASpringAnimationRef ref ) = CGFloat  
CASpringAnimationSetMass( CASpringAnimationRef ref, CGFloat value )  
CASpringAnimationSettlingDuration( CASpringAnimationRef ref ) = CFTimeInterval  
CASpringAnimationStiffness( CASpringAnimationRef ref ) = CGFloat  
CASpringAnimationSetStiffness( CASpringAnimationRef ref, CGFloat value )
```

### Apple documentation

[CASpringAnimation](#)



---

## CATextLayer

### Functions

Init

```
CATextLayerInit = CATextLayerRef// autoreleased
```

Get/set

```
CATextLayerString( CATextLayerRef layer ) = CTypeRef// CFStringRef, CFAttributedStringRef
```

```
CATextLayerSetString( CATextLayerRef layer, CTypeRef string )// CFStringRef, CFAttributedStringRef
```

Properties

```
CATextLayerFont( CATextLayerRef layer ) = CTypeRef// CTFontRef, CGFontRef, FontRef (NSFont), CFStringRef (font name)
```

```
CATextLayerSetFont( CATextLayerRef layer, CTypeRef font )// CTFontRef, CGFontRef, FontRef (NSFont), CFStringRef (font name)
```

```
CATextLayerFontSize( CATextLayerRef layer ) = CGFloat
```

```
CATextLayerSetFontSize( CATextLayerRef layer, CGFloat size )
```

```
CATextLayerForegroundColor( CATextLayerRef layer ) = ColorRef// macOS 10.8+
```

```
CATextLayerSetForegroundColor( CATextLayerRef layer, ColorRef col )// macOS 10.8+
```

Alignment/truncation

```
CATextLayerIsWrapped( CATextLayerRef layer ) = Boolean
```

```
CATextLayerSetWrapped( CATextLayerRef layer, Boolean flag )
```

```
CATextLayerAlignmentMode( CATextLayerRef layer ) = CFStringRef
```

```
CATextLayerSetAlignmentMode( CATextLayerRef layer, CFStringRef alignment )
```

```
CATextLayerTruncationMode( CATextLayerRef layer ) = CFStringRef
```

```
CATextLayerSetTruncationMode( CATextLayerRef layer, CFStringRef truncation )
```

Instance properties

```
CATextLayerAllowsFontSubpixelQuantization( CATextLayerRef layer ) = Boolean// macOS 10.11+
```

```
CATextLayerSetAllowsFontSubpixelQuantization( CATextLayerRef layer, Boolean flag )// macOS 10.11+
```

### Apple documentation

[CATextLayer](#)



---

## CATiledLayer

### Functions

Init

```
CATiledLayerInit = CATiledLayerRef// autoreleased
```

Visual fade

```
CATiledLayerFadeDuration = CFTimeInterval
```

Levels of detail

```
CATiledLayerLevelsOfDetail( CATiledLayerRef layer ) = NSUInteger
```

```
CATiledLayerSetLevelsOfDetail( CATiledLayerRef layer, NSUInteger levels )
```

```
CATiledLayerLevelsOfDetailBias( CATiledLayerRef layer ) = NSUInteger
```

```
CATiledLayerSetLevelsOfDetailBias( CATiledLayerRef layer, NSUInteger levels )
```

Tile size

```
CATiledLayerTileSize( CATiledLayerRef layer ) = CGSize
```

```
CATiledLayerSetTileSize( CATiledLayerRef layer, CGSize size )
```

### Apple documentation

[CATiledLayer](#)



---

## CATransform3D

### Functions

Create

```
CATransform3DMakeTranslation( CGFloat tx, CGFloat ty, CGFloat tz ) = CATransform3D
CATransform3DMakeScale( CGFloat sx, CGFloat sy, CGFloat sz ) = CATransform3D
CATransform3DMakeRotation( CGFloat angle, CGFloat x, CGFloat y, CGFloat z ) = CATransform3D
```

Applying

```
CATransform3DConcat( CATransform3D a, CATransform3D b ) = CATransform3D
CATransform3DTranslate( CATransform3D t, CGFloat tx, CGFloat ty, CGFloat tz ) = CATransform3D
CATransform3DScale( CATransform3D t, CGFloat sx, CGFloat sy, CGFloat sz ) = CATransform3D
CATransform3DRotate( CATransform3D t, CGFloat angle, CGFloat x, CGFloat y, CGFloat z ) = CATransform3D
CATransform3DInvert( CATransform3D t ) = CATransform3D
```

Properties

```
CATransform3DIsAffine( CATransform3D t ) = Boolean
CATransform3DIsIdentity( CATransform3D t ) = Boolean
CATransform3DEqualToTransform( CATransform3D a, CATransform3D b ) = Boolean
```

Convert from CG

```
CATransform3DMakeAffineTransform( CGAffineTransform m ) = CATransform3D
CATransform3DGetAffineTransform( CATransform3D t ) = CGAffineTransform
```

### Apple documentation

[CATransform3D](#)



---

### CATransformLayer

#### Functions

Init

```
CATransformLayerInit = CATransformLayerRef// autoreleased
```

#### Apple documentation

[CATransformLayer](#)



---

## CATransition

### Functions

Init

```
CATransitionAnimation = CATransitionRef
```

// Start and end point

```
CATransitionStartProgress( CATransitionRef ref ) = float  
CATransitionSetStartProgress( CATransitionRef ref, float value )  
CATransitionEndProgress( CATransitionRef ref ) = float  
CATransitionSetEndProgress( CATransitionRef ref, float value )
```

Properties

```
CATransitionType( CATransitionRef ref ) = CFStringRef  
CATransitionSetType( CATransitionRef ref, CFStringRef type )  
CATransitionSubtype( CATransitionRef ref ) = CFStringRef  
CATransitionSetSubtype( CATransitionRef ref, CFStringRef subtype )
```

Filter

```
CATransitionFilter( CATransitionRef ref ) = CFTypeRef  
CATransitionSetFilter( CATransitionRef ref, CFTypeRef filter )
```

### Apple documentation

[CATransition](#)



---

### CAValueFunction

#### Functions

Create

```
CAValueFunctionWithName( CFStringRef name ) = CAValueFunctionRef
```

Properties

```
CAValueFunctionName( CAValueFunctionRef ref ) = CFStringRef
```

#### Apple documentation

[CAValueFunction](#)



## CharacterSet

### Functions

#### Standard

```
CharacterSetAlphanumericSet = CFCharacterSetRef
CharacterSetCapitalizedLetterSet = CFCharacterSetRef
CharacterSetControlSet = CFCharacterSetRef
CharacterSetDecimalDigitSet = CFCharacterSetRef
CharacterSetDecomposableSet = CFCharacterSetRef
CharacterSetIllegalSet = CFCharacterSetRef
CharacterSetLetterSet = CFCharacterSetRef
CharacterSetLowercaseLetterSet = CFCharacterSetRef
CharacterSetNewlineSet = CFCharacterSetRef
CharacterSetNonBaseSet = CFCharacterSetRef
CharacterSetPunctuationSet = CFCharacterSetRef
CharacterSetSymbolSet = CFCharacterSetRef
CharacterSetUppercaseLetterSet = CFCharacterSetRef
CharacterSetWhitespaceAndNewlineSet = CFCharacterSetRef
CharacterSetWhitespaceSet = CFCharacterSetRef
```

#### URL encoding

```
CharacterSetURLFragmentAllowedSet = CFCharacterSetRef// macOS 10.9+
CharacterSetURLHostAllowedSet = CFCharacterSetRef// macOS 10.9+
CharacterSetURLPasswordAllowedSet = CFCharacterSetRef// macOS 10.9+
CharacterSetURLPathAllowedSet = CFCharacterSetRef// macOS 10.9+
CharacterSetURLQueryAllowedSet = CFCharacterSetRef// macOS 10.9+
CharacterSetURLUserAllowedSet = CFCharacterSetRef// macOS 10.9+
```

#### Custom character set

```
CharacterSetWithCoder( CoderRef coder ) = CFCharacterSetRef
CharacterSetWithCharactersInString( CFStringRef string ) = CFCharacterSetRef
CharacterSetWithRange( CFRange range ) = CFCharacterSetRef
```

#### Bitmap representations

```
CharacterSetWithBitmapRepresentation( CFDataRef data ) = CFCharacterSetRef
CharacterSetWithContentsOfURL( CFURLRef url ) = CFCharacterSetRef
CharacterSetBitmapRepresentation( CFCharacterSetRef ref ) = CFDataRef
```

#### Inverting

```
CharacterSetInvertedSet( CFCharacterSetRef set ) = CFCharacterSetRef
```

#### Testing membership

```
CharacterSetCharacterIsMember( CFCharacterSetRef set, unichar chr ) = Boolean
CharacterSetHasMemberInPlane( CFCharacterSetRef set, UInt8 plane ) = Boolean
CharacterSetIsSupersetOfSet( CFCharacterSetRef set, CFCharacterSetRef otherSet ) = Boolean
CharacterSetLongCharacterIsMember( CFCharacterSetRef set, UInt32 longChar ) = Boolean
```

```
// === mutable character set === //
```

#### Adding and removing characters

```
CharacterSetAddCharactersInRange( CFMutableCharacterSetRef set, CFRange range )
CharacterSetRemoveCharactersInRange( CFMutableCharacterSetRef set, CFRange range )
CharacterSetAddCharactersInString( CFMutableCharacterSetRef set, CFStringRef string )
CharacterSetRemoveCharactersInString( CFMutableCharacterSetRef set, CFStringRef string )
```

#### Combining sets

```
CharacterSetFormIntersectionWithCharacterSet( CFMutableCharacterSetRef set, CFCharacterSetRef otherSet )
CharacterSetFormUnionWithCharacterSet( CFMutableCharacterSetRef set, CFCharacterSetRef otherSet )
```

#### Inverting

```
CharacterSetInvert( CFMutableCharacterSetRef set )
```

### Apple documentation

[NSCharacterSet](#)

[NSMutableCharacterSet](#)





---

### CIContext

#### Functions

Create

```
CIContextWithOptions( CFDictionaryRef options ) = CIContextRef
```

Render images

```
CIContextCreateCGImage( CIContextRef ctx, CIImageRef image, CGRect r ) = CGImageRef
```

#### Apple documentation

[CIContext](#)



---

## CIDetector

### Functions

Create

```
CIDetectorOfType( CFStringRef type, CIContextRef context, CFDictionaryRef options ) = CIDetectorRef
```

Find features

```
CIDetectorFeaturesInImage( CIDetectorRef ref, CIImageRef image ) = CFArrayRef
```

```
CIDetectorFeaturesInImageWithOptions( CIDetectorRef ref, CIImageRef image, CFDictionaryRef options ) = CFArrayRef//  
macOS 10.8+
```

### Apple documentation

[CIDetector](#)



---

## CIFeature

### Functions

Properties

`CIFeatureBounds( CIFeatureRef ref ) = CGRect`

Type

`CIFeatureType( CIFeatureRef ref ) = CFStringRef`

### Apple documentation

[CIFeature](#)



---

## CIFilter

### Functions

Create

```
CIFilterWithName( CFStringRef name ) = CIFilterRef  
CIFilterWithNameAndInputParameters( CFStringRef name, CFDictionaryRef params ) = CIFilterRef
```

Create from raw image

```
CIFilterWithImageData( CFDataRef dta, CFDictionaryRef options ) = CIFilterRef  
CIFilterWithURL( CFURLRef url, CFDictionaryRef options ) = CIFilterRef
```

Registered filters

```
CIFilterFilterNamesInCategories( CFArrayRef categories ) = CFArrayRef  
CIFilterFilterNamesInCategory( CFStringRef category ) = CFArrayRef
```

Parameters and attributes

```
CIFilterSetName( CIFilterRef ref, CFStringRef name )  
CIFilterIsEnabled( CIFilterRef ref ) = Boolean  
CIFilterAttributes( CIFilterRef ref ) = CFDictionaryRef  
CIFilterInputKeys( CIFilterRef ref ) = CFArrayRef  
CIFilterOutputKeys( CIFilterRef ref ) = CFArrayRef  
CIFilterOutputImage( CIFilterRef ref ) = CIImageRef// macOS 10.10+
```

Default values

```
CIFilterSetDefaults( CIFilterRef ref )
```

Localized info for registered filters

```
CIFilterLocalizedStringForFilterName( CFStringRef filterName ) = CFStringRef  
CIFilterLocalizedStringForCategory( CFStringRef category ) = CFStringRef  
CIFilterLocalizedStringForFilterName( CFStringRef filterName ) = CFStringRef  
CIFilterLocalizedStringReferenceDocumentationForFilterName( CFStringRef filterName ) = CFURLRef
```

Serialize and deserialize

```
CIFilterSerializedXMPFromFilters( CFArrayRef filters, CGRect extent ) = CFDataRef// macOS 10.9+  
CIFilterFilterArrayFromSerializedXMP( CFDataRef xmpData, CGRect extent, ErrorRef *err ) = CFArrayRef// err can be  
NULL // macOS 10.9+
```

### Apple documentation

[CIFilter](#)



## CIImage

### Functions

#### Create

```
CIImageEmptyImage( CFStringRef name ) = CIImageRef
CIImageWithCGImage( CGImageRef image ) = CIImageRef
CIImageWithCGImageAndOptions( CGImageRef image, CFDictionaryRef options ) = CIImageRef
CIImageWithContentsOfURL( CFURLRef url ) = CIImageRef
CIImageWithData( CFDataRef dta ) = CIImageRef
CIImageWithDataAndOptions( CFDataRef dta, CFDictionaryRef options ) = CIImageRef
```

#### Create by modifying existing image

```
CIImageByApplyingFilterWithInputParameters( CIImageRef ref, CFStringRef filterName, CFDictionaryRef params ) =
CIImageRef// macOS 10.10+
CIImageByApplyingFilter( CIImageRef ref, CFStringRef filterName ) = CIImageRef// macOS 10.13+
CIImageByApplyingTransform( CIImageRef ref, long orientation ) = CIImageRef
CIImageByCroppingToRect( CIImageRef ref, CGRect r ) = CIImageRef
CIImageByApplyingOrientation( CIImageRef ref, long orientation ) = CIImageRef
CIImageByClampingToExtent( CIImageRef ref ) = CIImageRef
CIImageByClampingToRect( CIImageRef ref, CGRect r ) = CIImageRef
CIImageByCompositingOverImage( CIImageRef ref, CIImageRef otherImage ) = CIImageRef
CIImageByPremultiplyingAlpha( CIImageRef ref ) = CIImageRef
CIImageByUnpremultiplyingAlpha( CIImageRef ref ) = CIImageRef
CIImageBySettingAlphaOneInExtent( CIImageRef ref, CGRect extent ) = CIImageRef
CIImageByApplyingGaussianBlurWithSigma( CIImageRef ref, double sigma ) = CIImageRef
CIImageBySettingProperties( CIImageRef ref, CFDictionaryRef properties ) = CIImageRef
```

#### Info

```
CIImageExtent( CIImageRef ref ) = CGRect
CIImageProperties( CIImageRef ref ) = CFDictionaryRef
CIImageURL( CIImageRef ref ) = CFURLRef
CIImageTransformForOrientation( CIImageRef ref, long orientation ) = CGAffineTransform
```

#### Drawing

```
CIImageDrawAtPoint( CIImageRef ref, CGPoint pt, CGRect fromRect, NSCompositingOperation op, CGFloat fraction )
CIImageDrawInRect( CIImageRef ref, CGRect r, CGRect fromRect, NSCompositingOperation op, CGFloat fraction )
```

#### Autoadjustment filters

```
CIImageAutoAdjustmentFilters( CIImageRef ref ) = CFArrayRef
CIImageAutoAdjustmentFiltersWithOptions( CIImageRef ref, CFDictionaryRef options ) = CFArrayRef
```

#### Filter regions of interest

```
CIImageRegionOfInterestForImage( CIImageRef ref, CIImageRef otherImage, CGRect r ) = CGRect
```

### Apple documentation

[CIImage](#)



---

### CIImageRep

#### Functions

Init

```
CIImageRepWithCIImage( CIImageRef image ) = CIImageRepRef
```

Return image

```
CIImageRepCIImage( CIImageRepRef ref ) = CIImageRef
```

#### Apple documentation

[NSCIImageRep](#)



---

## CIQRCodeFeature

### Functions

Locate detected feature

```
CIQRCodeFeatureBounds( CIQRCodeFeatureRef ref ) = CGRect
```

Decode barcode

```
CIQRCodeFeatureMessageString( CIQRCodeFeatureRef ref ) = CFStringRef
```

```
CIQRCodeFeatureSymbolDescriptor( CIQRCodeFeatureRef ref ) = CIQRCodeDescriptorRef// macOS 10.13+
```

Identify corners of barcode

```
CIQRCodeFeatureBottomLeft( CIQRCodeFeatureRef ref ) = CGPoint
```

```
CIQRCodeFeatureBottomRight( CIQRCodeFeatureRef ref ) = CGPoint
```

```
CIQRCodeFeatureTopLeft( CIQRCodeFeatureRef ref ) = CGPoint
```

```
CIQRCodeFeatureTopRight( CIQRCodeFeatureRef ref ) = CGPoint
```



---

## ClickGestureRecognizer

### Functions

Init

```
ClickGestureRecognizerInit( ptr callback, ptr userData ) = ClickGestureRecognizerRef// autoreleased
```

Configure

```
ClickGestureRecognizerButtonMask( ClickGestureRecognizerRef ref ) = NSUInteger
```

```
ClickGestureRecognizerSetButtonMask( ClickGestureRecognizerRef ref, NSUInteger mask )
```

```
ClickGestureRecognizerNumberOfClicksRequired( ClickGestureRecognizerRef ref ) = NSUInteger
```

```
ClickGestureRecognizerSetNumberOfClicksRequired( ClickGestureRecognizerRef ref, NSUInteger clicks )
```

Instance properties

```
ClickGestureRecognizerNumberOfTouchesRequired( ClickGestureRecognizerRef ref ) = NSUInteger// mac OS 10.12.2
```

```
ClickGestureRecognizerSetNumberOfTouchesRequired( ClickGestureRecognizerRef ref, NSUInteger touches )// mac OS 10.12.2
```

### Apple documentation

[NSClickGestureRecognizer](#)





---

## ClipView

### Functions

Document view

```
ClipViewDocumentView( ClipViewRef ref ) = ViewRef
ClipViewSetDocumentView( ClipViewRef ref, ViewRef docView )
```

Scrolling

```
ClipViewScrollToPoint( ClipViewRef ref, CGPoint pt )
ClipViewConstrainBoundsRect( ClipViewRef ref, CGRect r )// macOS 10.9+
```

Scrolling efficiency

```
ClipViewCopiesOnScroll( ClipViewRef ref ) = Boolean
ClipViewSetCopiesOnScroll( ClipViewRef ref, Boolean flag )
```

Content insets

```
ClipViewContentInsets( ClipViewRef ref ) = UIEdgeInsets// macOS 10.10+
ClipViewSetContentInsets( ClipViewRef ref, UIEdgeInsets insets )// macOS 10.10+
ClipViewAutomaticallyAdjustsContentInsets( ClipViewRef ref ) = Boolean// macOS 10.10+
ClipViewSetAutomaticallyAdjustsContentInsets( ClipViewRef ref, Boolean flag )// macOS 10.10+
```

Visible portion

```
ClipViewDocumentRect( ClipViewRef ref ) = CGRect
ClipViewDocumentVisibleRect( ClipViewRef ref ) = CGRect
```

Document cursor

```
ClipViewDocumentCursor( ClipViewRef ref ) = CursorRef
ClipViewSetDocumentCursor( ClipViewRef ref, CursorRef curs )
```

Background color

```
ClipViewDrawsBackground( ClipViewRef ref ) = Boolean
ClipViewSetDrawsBackground( ClipViewRef ref, Boolean flag )
ClipViewBackgroundColor( ClipViewRef ref ) = ColorRef
ClipViewSetBackgroundColor( ClipViewRef ref, ColorRef col )
```

Custom

```
ClipViewBounds( ClipViewRef ref ) = CGRect
ClipViewSetBounds( ClipViewRef ref, CGRect r )
```

### Apple documentation

[NSClipView](#)



---

## CMTime

### Functions

Create

```
CMTimeMake( SInt64 value, SInt32 timescale ) = CMTime
CMTimeMakeFromDictionary( CFDictionaryRef dict ) = CMTime
CMTimeMakeWithEpoch( SInt64 value, SInt32 timescale, SInt64 epoch ) = CMTime
CMTimeMakeWithSeconds( double second, SInt32 preferredTimescale ) = CMTime
```

Common operations

```
CMTimeAdd( CMTime addend1, CMTime addend2 ) = CMTime
CMTimeSubtract( CMTime minuend, CMTime subtrahend ) = CMTime
CMTimeMultiply( CMTime time, SInt32 multiplier ) = CMTime
CMTimeMultiplyByFloat64( CMTime time, double multiplier ) = CMTime
CMTimeMultiplyByRatio( CMTime time, SInt32 multiplier, SInt32 divisor ) = CMTime // macOS 10.10+
CMTimeConvertScale( CMTime time, SInt32 newTimescale, CMTimeRoundingMethod method ) = CMTime
```

Evaluating times

```
CMTimeCompare( CMTime time1, CMTime time2 ) = SInt32
CMTimeAbsoluteValue( CMTime time ) = CMTime
CMTimeGetSeconds( CMTime time ) = double
CMTimeMaximum( CMTime time1, CMTime time2 ) = CMTime
CMTimeMinimum( CMTime time1, CMTime time2 ) = CMTime
```

```
CMTIME_IS_VALID( CMTime time ) = Boolean
CMTIME_IS_INVALID( CMTime time ) = Boolean
CMTIME_IS_POSITIVE_INFINITY( CMTime time ) = Boolean
CMTIME_IS_NEGATIVE_INFINITY( CMTime time ) = Boolean
CMTIME_IS_INDEFINITE( CMTime time ) = Boolean
CMTIME_IS_NUMERIC( CMTime time ) = Boolean
CMTIME_IS_HAS_BEEN_ROUNDED( CMTime time ) = Boolean
```

Utilities

```
CMTimeShow( CMTime time )
CMTimeCopyDescription( CFAllocatorRef allocator, CMTime time ) = CFStringRef // must be released by the caller
CMTimeCopyAsDictionary( CMTime time, CFAllocatorRef allocator ) = CFDictionaryRef // must be released by the caller
```

### Apple documentation

[CMTime](#)



---

## Coder

### Functions

Decoding

`CoderDecodeObjectOfClass( CoderRef ref, ClassRef class, CFStringRef key ) = CFTypeRef` // macOS 10.8+

### Apple documentation

[NSCoder](#)



## Color

### Functions

UI element

- Label

```
ColorLabel = ColorRef// macOS 10.10+
ColorSecondaryLabel = ColorRef// macOS 10.10+
ColorTertiaryLabel = ColorRef// macOS 10.10+
ColorQuaternaryLabel = ColorRef// macOS 10.10+
```

- Text

```
ColorText = ColorRef
ColorPlaceholderText = ColorRef// macOS 10.10+
ColorSelectedText = ColorRef
ColorTextBackground = ColorRef
ColorSelectedTextBackground = ColorRef
ColorKeyboardFocusIndicator = ColorRef
ColorUnemphasizedSelectedText = ColorRef// macOS 10.14+
ColorUnemphasizedSelectedTextBackground = ColorRef// macOS 10.14+
```

- Content

```
ColorLink = ColorRef// macOS 10.10+
ColorSeparator = ColorRef// macOS 10.14+
ColorSelectedContentBackground = ColorRef// macOS 10.14+
ColorUnemphasizedSelectedContentBackground = ColorRef// macOS 10.14+
```

- Menu

```
ColorSelectedMenuItemText = ColorRef
```

- Table

```
ColorGrid = ColorRef
ColorHeaderText = ColorRef
ColorAlternatingContentBackgrounds = CFArrayRef// macOS 10.14+
```

- Control

```
ColorControlAccent = ColorRef// macOS 10.14+
ColorControl = ColorRef
ColorControlBackground = ColorRef
ColorControlText = ColorRef
ColorDisabledControlText = ColorRef
ColorCurrentControlTint = NSControlTint
ColorSelectedControl = ColorRef
ColorSelectedControlText = ColorRef
ColorAlternateSelectedControlText = ColorRef
ColorScrubberTexturedBackground = ColorRef// macOS 10.11+
```

- Window

```
ColorWindowBackground = ColorRef
ColorWindowFrameText = ColorRef
ColorUnderPageBackground = ColorRef// macOS 10.8+
```

- Highlights and shadows

```
ColorFindHighlight = ColorRef// macOS 10.13+
ColorHighlight = ColorRef
ColorShadow = ColorRef
```

- Deprecated

```
ColorAlternateSelectedControl = ColorRef
ColorControlAlternatingRowBackground = CFArrayRef
ColorControlHighlight = ColorRef
ColorControlShadow = ColorRef
ColorControlDarkShadow = ColorRef
ColorHeader = ColorRef
ColorKnob = ColorRef
ColorScrollBar = ColorRef
ColorSecondarySelectedControl = ColorRef
ColorSelectedKnob = ColorRef
ColorSelectedMenuItem = ColorRef
ColorWindowFrame = ColorRef
```

## System

### - Adaptable

```
ColorSystemBlue = ColorRef// macOS 10.11+
ColorSystemBrown = ColorRef// macOS 10.11+
ColorSystemGray = ColorRef// macOS 10.11+
ColorSystemGreen = ColorRef// macOS 10.11+
ColorSystemOrange = ColorRef// macOS 10.11+
ColorSystemPink = ColorRef// macOS 10.11+
ColorSystemPurple = ColorRef// macOS 10.11+
ColorSystemRed = ColorRef// macOS 10.11+
ColorSystemYellow = ColorRef// macOS 10.11+
```

### - Transparent

```
ColorClear = ColorRef
```

### - Fixed

```
ColorBlack = ColorRef
ColorBlue = ColorRef
ColorBrown = ColorRef
ColorCyan = ColorRef
ColorDarkGray = ColorRef
ColorGray = ColorRef
ColorGreen = ColorRef
ColorLightGray = ColorRef
ColorMagenta = ColorRef
ColorOrange = ColorRef
ColorPurple = ColorRef
ColorRed = ColorRef
ColorWhite = ColorRef
ColorYellow = ColorRef
```

## Creation

### - Asset catalogs

```
ColorNamed( CFStringRef name ) = ColorRef// macOS 10.13+
ColorNamedBundle( CFStringRef name, BundleRef bundle ) = ColorRef// macOS 10.13+
ColorWithCatalogName( CFStringRef catalogueName, CFStringRef colorName ) = ColorRef
```

### - System tint

```
ColorForControlTint( NSControlTint tint ) = ColorRef
```

### - RGB

```
ColorWithSRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha ) = ColorRef
ColorWithDisplayP3( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha ) = ColorRef// macOS 10.12+
ColorWithRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha ) = ColorRef// macOS 10.9+
ColorWithCalibratedRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha ) = ColorRef
ColorWithDeviceRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha ) = ColorRef
```

### - HSB

```
ColorWithCalibratedHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha ) = ColorRef
ColorWithDeviceHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha ) = ColorRef
ColorWithHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha ) = ColorRef// macOS 10.9+
```

### - CMYK

```
ColorWithDeviceCMYK( CGFloat cyan, CGFloat magenta, CGFloat yellow, CGFloat black, CGFloat alpha ) = ColorRef
```

### - White

```
ColorWithWhite( CGFloat white, CGFloat alpha ) = ColorRef// macOS 10.9+
ColorWithCalibratedWhite( CGFloat white, CGFloat alpha ) = ColorRef
ColorWithDeviceWhite( CGFloat white, CGFloat alpha ) = ColorRef
ColorWithGenericGamma22White( CGFloat white, CGFloat alpha ) = ColorRef
```

### - Pattern

```
ColorWithPatternImage( ptr image ) = ColorRef
ColorPatternImage( ColorRef col ) = ImageRef
```

### - Arbitrary color space

```
ColorWithColorSpace( ColorSpaceRef cs, const CGFloat *components, NSInteger count ) = ColorRef
```

### - Other types

```
ColorWithCGColor( CGColorRef col ) = ColorRef// macOS 10.8+
ColorWithCIColor( CIColorRef col ) = ColorRef
```

## Appearance

```
ColorWithSystemEffect( ColorRef col, NSColorSystemEffect effect ) = ColorRef// macOS 10.14+
```

## Transforming

```
ColorUsingColorSpace( ColorRef col, ColorSpaceRef cs ) = ColorRef
ColorBlendedWithFractionOfColor( ColorRef col1, CGFloat fraction, ColorRef col2 ) = ColorRef
ColorWithAlphaComponent( ColorRef col, CGFloat alpha ) = ColorRef
ColorHighlightWithLevel( ColorRef col, CGFloat value ) = ColorRef
```

```
ColorShadowWithLevel( ColorRef col, CGFloat value ) = ColorRef
```

#### Alpha support

```
ColorIgnoresAlpha = Boolean
```

#### Copy paste color info

```
ColorFromPasteboard( CocoaPasteboardRef pb ) = ColorRef  
ColorWriteToPasteboard( ColorRef col, CocoaPasteboardRef pb )
```

#### Component values

```
ColorGetCMYK( ColorRef col, CGFloat *c, CGFloat *m, CGFloat *y, CGFloat *k, CGFloat *a )  
ColorGetHSB( ColorRef col, CGFloat *h, CGFloat *s, CGFloat *b, CGFloat *a )  
ColorGetRGB( ColorRef col, CGFloat *r, CGFloat *g, CGFloat *b, CGFloat *a )  
ColorGetWhite( ColorRef col, CGFloat *w, CGFloat *a )  
ColorNumberOfComponents( ColorRef col ) = NSInteger  
ColorGetComponents( ColorRef col, CGFloat *components )
```

#### Individual components

```
ColorAlphaComponent( ColorRef col ) = CGFloat  
ColorWhiteComponent( ColorRef col ) = CGFloat  
ColorRedComponent( ColorRef col ) = CGFloat  
ColorGreenComponent( ColorRef col ) = CGFloat  
ColorBlueComponent( ColorRef col ) = CGFloat  
ColorCyanComponent( ColorRef col ) = CGFloat  
ColorMagentaComponent( ColorRef col ) = CGFloat  
ColorYellowComponent( ColorRef col ) = CGFloat  
ColorBlackComponent( ColorRef col ) = CGFloat  
ColorHueComponent( ColorRef col ) = CGFloat  
ColorSaturationComponent( ColorRef col ) = CGFloat  
ColorBrightnessComponent( ColorRef col ) = CGFloat  
ColorCatalogNameComponent( ColorRef col ) = CFStringRef  
ColorLocalizedCatalogNameComponent( ColorRef col ) = CFStringRef  
ColorNameComponent( ColorRef col ) = CFStringRef  
ColorLocalizedNameComponent( ColorRef col ) = CFStringRef
```

#### Color space

```
ColorType( ColorRef col ) = NSColorType// macOS 10.13+  
ColorUsingType( ColorRef col, NSColorType type) = ColorRef// macOS 10.13+  
ColorColorSpace( ColorRef col ) = ColorSpaceRef
```

#### CGColor

```
ColorCGColor( ColorRef col ) = CGColorRef// macOS 10.8+
```

#### Drawing

```
ColorDrawSwatchInRect( ColorRef col, CGRect r )  
ColorSet( ColorRef col )  
ColorSetFill( ColorRef col )  
ColorSetStroke( ColorRef col )
```

#### Deprecated

```
ColorColorSpaceName( ColorRef col ) = CFStringRef
```

#### Convenience

```
ColorSetBlack  
ColorSetBlue  
ColorSetBrown  
ColorSetClear  
ColorSetCyan  
ColorSetDarkGray  
ColorSetGray  
ColorSetGreen  
ColorSetLightGray  
ColorSetMagenta  
ColorSetOrange  
ColorSetPurple  
ColorSetRed  
ColorSetWhite  
ColorSetYellow
```

```
ColorSetWithCalibratedHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha )  
ColorSetWithDeviceHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha )  
ColorSetWithHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha )macOS 10.9+  
ColorSetWithSRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )  
ColorSetWithDisplayP3( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )macOS 10.12+  
ColorSetWithRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )macOS 10.9+  
ColorSetWithCalibratedRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )  
ColorSetWithDeviceRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )  
ColorSetWithWhite( CGFloat white, CGFloat alpha )macOS 10.9+  
ColorSetWithCalibratedWhite( CGFloat white, CGFloat alpha )  
ColorSetWithDeviceWhite( CGFloat white, CGFloat alpha )  
ColorSetWithGenericGamma22White( CGFloat white, CGFloat alpha )
```

```
ColorSetWithDeviceCMYK( CGFloat cyan, CGFloat magenta, CGFloat yellow, CGFloat black, CGFloat alpha )
```

```
ColorSetFillBlack
ColorSetFillBlue
ColorSetFillBrown
ColorSetFillClear
ColorSetFillCyan
ColorSetFillDarkGray
ColorSetFillGray
ColorSetFillGreen
ColorSetFillLightGray
ColorSetFillMagenta
ColorSetFillOrange
ColorSetFillPurple
ColorSetFillRed
ColorSetFillWhite
ColorSetFillYellow
ColorSetFillWithCalibratedHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha )
ColorSetFillWithDeviceHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha )
ColorSetFillWithHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha )macOS 10.9+
ColorSetFillWithSRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )
ColorSetFillWithDisplayP3( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )macOS 10.12+
ColorSetFillWithRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )macOS 10.9+
ColorSetFillWithCalibratedRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )
ColorSetFillWithDeviceRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )
ColorSetFillWithWhite( CGFloat white, CGFloat alpha )macOS 10.9+
ColorSetFillWithCalibratedWhite( CGFloat white, CGFloat alpha )
ColorSetFillWithDeviceWhite( CGFloat white, CGFloat alpha )
ColorSetFillWithGenericGamma22White( CGFloat white, CGFloat alpha )
ColorSetFillWithDeviceCMYK( CGFloat cyan, CGFloat magenta, CGFloat yellow, CGFloat black, CGFloat alpha )
```

```
ColorSetStrokeBlack
ColorSetStrokeBlue
ColorSetStrokeBrown
ColorSetStrokeClear
ColorSetStrokeCyan
ColorSetStrokeDarkGray
ColorSetStrokeGray
ColorSetStrokeGreen
ColorSetStrokeLightGray
ColorSetStrokeMagenta
ColorSetStrokeOrange
ColorSetStrokePurple
ColorSetStrokeRed
ColorSetStrokeWhite
ColorSetStrokeYellow
```

```
ColorSetStrokeWithCalibratedHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha )
ColorSetStrokeWithDeviceHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha )
ColorSetStrokeWithHSB( CGFloat hue, CGFloat saturation, CGFloat brightness, CGFloat alpha )macOS 10.9+
ColorSetStrokeWithSRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )
ColorSetStrokeWithDisplayP3( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )macOS 10.12+
ColorSetStrokeWithRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )macOS 10.9+
ColorSetStrokeWithCalibratedRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )
ColorSetStrokeWithDeviceRGB( CGFloat red, CGFloat green, CGFloat blue, CGFloat alpha )
ColorSetStrokeWithWhite( CGFloat white, CGFloat alpha )macOS 10.9+
ColorSetStrokeWithCalibratedWhite( CGFloat white, CGFloat alpha )
ColorSetStrokeWithDeviceWhite( CGFloat white, CGFloat alpha )
ColorSetStrokeWithGenericGamma22White( CGFloat white, CGFloat alpha )
ColorSetStrokeWithDeviceCMYK( CGFloat cyan, CGFloat magenta, CGFloat yellow, CGFloat black, CGFloat alpha )
```

## Apple documentation

[NSColor](#)



---

## ColorList

### Functions

Init

```
ColorListWithName( CFStringRef name ) = ColorListRef
ColorListFromFileAtURL( CFStringRef name, CFURLRef url ) = ColorListRef
```

Lists

```
ColorListAvailableColorLists = CFArrayRef
ColorListNamed( CFStringRef name ) = ColorListRef
```

Properties

```
ColorListName( ColorListRef cl ) = CFStringRef
ColorListIsEditable( ColorListRef cl ) = Boolean
```

Managing colors

```
ColorListAllKeys( ColorListRef cl ) = CFArrayRef
ColorListColorWithKey( ColorListRef cl, CFStringRef key ) = ColorRef
ColorListInsertColor( ColorListRef cl, ColorRef col, CFStringRef key, NSUInteger index )
ColorListRemoveColorWithKey( ColorListRef cl, CFStringRef key )
ColorListSetColor( ColorListRef cl, ColorRef col, CFStringRef key )
```

Write and remove ColorList files

```
ColorListRemoveFile( ColorListRef cl )
ColorListWriteToURL( ColorListRef cl, CFURLRef url, ErrorRef *err ) = Boolean
```

Convenience

```
ColorListByAddingColor( ColorListRef cl, ColorRef col, CFStringRef key ) = ColorListRef
ColorListByAddingColorList( ColorListRef cl1, ColorListRef cl2 ) = ColorListRef
```

### Apple documentation

[NSColorList](#)





---

## ColorPanel

### Description

The shared color panel.

### Functions

The shared color panel

```
ColorPanelShared = ColorPanelRef
```

Configure

```
ColorPanelAccessoryView = ViewRef
ColorPanelSetAccessoryView( ViewRef ref )
ColorPanelIsContinuous = Boolean
ColorPanelSetContinuous( Boolean flag )
ColorPanelMode = NSColorPanelMode
ColorPanelSetMode( NSColorPanelMode mode )
ColorPanelSetActionCallback( ptr callback, ptr userData )
ColorPanelShowsAlpha = Boolean
ColorPanelSetShowsAlpha( Boolean flag )
```

Color lists

```
ColorPanelAttachColorList( ColorListRef cl )
ColorPanelDetachColorList( ColorListRef cl )
```

Picker modes

```
ColorPanelSetPickerMask( NSColorPanelOptions options )
ColorPanelSetPickerMode( NSInteger mode )
```

Color

```
ColorPanelColor = ColorRef
ColorPanelSetColor( ColorRef col )
```

Info

```
ColorPanelAlpha = CGFloat
```

Convenience

```
ColorPanelShow
ColorPanelClose
```

### Apple documentation

[NSColorPanel](#)



---

## ColorList

### Functions

Named spaces

```
ColorSpaceDeviceRGBSpace( ptr ) = ColorSpaceRef
ColorSpaceGenericRGBSpace( ptr ) = ColorSpaceRef
ColorSpaceDeviceCMYKSpace( ptr ) = ColorSpaceRef
ColorSpaceGenericCMYKSpace( ptr ) = ColorSpaceRef
ColorSpaceDeviceGraySpace( ptr ) = ColorSpaceRef
ColorSpaceGenericGraySpace( ptr ) = ColorSpaceRef
ColorSpaceSRGBSpace( ptr ) = ColorSpaceRef
ColorSpaceExtendedSRGBSpace( ptr ) = ColorSpaceRef// macOS 10.12+
ColorSpaceP3Space( ptr ) = ColorSpaceRef// macOS 10.12+
ColorSpaceGenericGamma22GraySpace( ptr ) = ColorSpaceRef
ColorSpaceExtendedGenericGamma22GraySpace( ptr ) = ColorSpaceRef// macOS 10.12+
ColorSpaceAdobeRGB1998Space( ptr ) = ColorSpaceRef
```

Available system spaces

```
ColorSpaceAvailableSpacesWithModel( NSColorSpaceModel model ) = CFArrayRef
```

Custom spaces

```
ColorSpaceWithCGColorSpace( CGColorSpaceRef cgColorSpace ) = ColorSpaceRef
ColorSpaceWithColorSyncProfile( ptr prof ) = ColorSpaceRef
ColorSpaceWithICCProfileData( CFDataRef dta ) = ColorSpaceRef
```

Data and attributes

```
ColorSpaceCGColorSpace( ColorSpaceRef ref ) = CGColorSpaceRef
ColorSpaceModel( ColorSpaceRef ref ) = NSColorSpaceModel
ColorSpaceColorSyncProfile( ColorSpaceRef ref ) = ptr
ColorSpaceICCProfileData( ColorSpaceRef ref ) = CFDataRef
ColorSpaceLocalizedString( ColorSpaceRef ref ) = CFStringRef
ColorSpaceNumberOfColorComponents( ColorSpaceRef ref ) = NSInteger
```

### Apple documentation

[NSColorSpace](#)



ColorWell

statement

**Syntax**  
`colorwell tag, enabled, color, rect, bordered, wndTag`

**Description**  
The **colorwell** statement puts a new colorwell in the current output cocoa window, or alters an existing colorwell's characteristics.

Parameters	
<i>tag</i>	A number to identify the colorwell. A negative tag hides the colorwell.
<i>enabled</i>	Enables or disables the colorwell
<i>color</i>	The color displayed in the colorwell. This param is a ColorRef (NSColor).
<i>rect</i>	Origin and size of the colorwell in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) CGRect value
<i>bordered</i>	A boolean value to indicate if the colorwell should have a border.
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

**Functions**  
`ColorWellWithTag( NSInteger tag ) = ColorWellRef`  
`ColorWellExists( NSInteger tag ) = Boolean`

Init  
`ColorWellInit( NSInteger tag, CGRect r ) = ColorWellRef// autoreleased`

Color  
`ColorWellColor( NSInteger tag ) = ColorRef`  
`ColorWellSetColor( NSInteger tag, ColorRef col )`  
`ColorWellTakeColorFrom( NSInteger colorwellTag, NSInteger otherViewTag )`

Activating  
`ColorWellActivate( NSInteger tag, Boolean exclusive )`  
`ColorWellIsActive( NSInteger tag ) = Boolean`  
`ColorWellDeactivate( NSInteger tag )`

Border  
`ColorWellIsBordered( NSInteger tag ) = Boolean`  
`ColorWellSetBordered( NSInteger tag, Boolean flag )`

Drawing  
`ColorWellDrawWellInside( NSInteger tag, CGRect insideRect )`

**See Also**  
[View](#)

**Apple documentation**  
[NSColorWell](#)



## ComboBox

statement

### Syntax

**combobox** *tag, enabled, items, text, rect, wndTag*

### Description

The **combobox** statement puts a new combobox in the current output cocoa window, or alters an existing combobox's characteristics.

### Parameters

<i>tag</i>	A number to identify the combobox. A negative tag hides the combobox
<i>enabled</i>	Enables or disables the combobox
<i>items</i>	A list of items to add to the combobox. Can be a CFArray or semicolon-delimited string of items.
<i>text</i>	The text displayed in the combobox.
<i>rect</i>	Origin and size of the combobox in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

Event	Description
<code>_textFieldXxxx</code>	See <a href="#">textfield</a> .
<code>_comboBoxSelectionDidChange</code>	The pop-up list selection has finished changing.
<code>_comboBoxSelectionIsChanging</code>	The pop-up list selection is changing.
<code>_comboBoxWillDismiss</code>	The pop-up list is about to be dismissed.
<code>_comboBoxWillPopUp</code>	The pop-up list is about to be displayed.

### Functions

```
ComboBoxWithTag( NSInteger tag ) = ComboBoxRef
```

```
ComboBoxExists( NSInteger tag ) = Boolean
```

Init

```
ComboBoxInit( NSInteger tag, CGRect r ) = ComboBoxRef// autoreleased
```

Display attributes

```
ComboBoxHasVerticalScroller( NSInteger tag ) = Boolean
```

```
ComboBoxSetHasVerticalScroller( NSInteger tag, Boolean flag )
```

```
ComboBoxInterCellSpacing( NSInteger tag ) = CGSize
```

```
ComboBoxSetInterCellSpacing( NSInteger tag, CGSize spacing )
```

```
ComboBoxIsBordered( NSInteger tag ) = Boolean
```

```
ComboBoxSetBordered( NSInteger tag, Boolean flag )
```

```
ComboBoxItemHeight( NSInteger tag ) = CGFloat
```

```
ComboBoxSetItemHeight( NSInteger tag, CGFloat height )
```

```
ComboBoxNumberOfVisibleItems( NSInteger tag ) = NSInteger
```

```
ComboBoxSetNumberOfVisibleItems( NSInteger tag, NSInteger value )
```

Data source

```
ComboBoxSetUsesDataSource( NSInteger tag, Boolean flag )
```

Configure

```
ComboBoxAddItems( NSInteger tag, CFArrayRef items )
```

```
ComboBoxAddItem( NSInteger tag, CTypeRef item )
```

```
ComboBoxInsertItem( NSInteger tag, CTypeRef item, NSInteger index )
```

```
ComboBoxValues( NSInteger tag ) = CFArrayRef
```

```
ComboBoxRemoveAllItems( NSInteger tag )
```

```
ComboBoxRemoveItemAtIndex( NSInteger tag, NSInteger index )
```

```
ComboBoxRemoveItemWithValue( NSInteger tag, CTypeRef value )
ComboBoxNumberOfItems( NSInteger tag ) = NSInteger
```

#### Displayed list

```
ComboBoxIndexOfItemWithValue( NSInteger tag, CTypeRef value ) = NSInteger
ComboBoxItemValueAtIndex( NSInteger tag, NSInteger index ) = CTypeRef
ComboBoxNoteNumberOfItemsChanged( NSInteger tag )
ComboBoxReloadData( NSInteger tag )
ComboBoxScrollItemAtIndexToTop( NSInteger tag, NSInteger index )
ComboBoxScrollItemAtIndexToVisible( NSInteger tag, NSInteger index )
```

#### Selection

```
ComboBoxDeselectItemAtIndex( NSInteger tag, NSInteger index )
ComboBoxIndexOfSelectedItem( NSInteger tag ) = NSInteger
ComboBoxValueOfSelectedItem( NSInteger tag ) = CTypeRef
ComboBoxSelectItemAtIndex( NSInteger tag, NSInteger index )
ComboBoxSelectItemWithValue( NSInteger tag, CTypeRef value )
```

#### Completing the text field

```
ComboBoxSetCompletes( NSInteger tag, Boolean flag )
```

#### Convenience

```
ComboBoxData( NSInteger tag ) = CFMutableArrayRef
ComboBoxSetData( NSInteger tag, CFMutableArrayRef array )
```

### See Also

[Control](#), [View](#)

### Apple documentation

[NSComboBox](#)



---

## ComparisonPredicate

### Functions

#### Create

```
ComparisonPredicateWithCustomSelector( ExpressionRef leftExpression, ExpressionRef rightExpression, CFStringRef  
customSelector ) = ComparisonPredicateRef  
ComparisonPredicateWithModifier( ExpressionRef leftExpression, ExpressionRef rightExpression,  
NSComparisonPredicateModifier modifier, NSComparisonPredicateOptions options ) = ComparisonPredicateRef
```

#### Info

```
ComparisonPredicateModifier( ComparisonPredicateRef ref ) = NSComparisonPredicateModifier  
ComparisonPredicateRightExpression( ComparisonPredicateRef ref ) = ExpressionRef  
ComparisonPredicateLeftExpression( ComparisonPredicateRef ref ) = ExpressionRef  
ComparisonPredicateOptions( ComparisonPredicateRef ref ) = NSComparisonPredicateOptions  
ComparisonPredicateOperatorType( ComparisonPredicateRef ref ) = NSPredicateOperatorType
```

### Apple documentation

[NSComparisonPredicate](#)



---

## Compiler

### Description

Send messages to FBtoC.

### Functions

```
CompilerRunFileAtURL( CFURLRef url )
CompilerAnalyzeFileAtURL( CFURLRef url )
CompilerBuildFileAtURL( CFURLRef url )
CompilerAbort
CompilerShowSettings
CompilerTerminate
CompilerHeadersURL = CFURLRef
```



---

### CompoundPredicate

#### Functions

##### Constructors

```
CompoundPredicateAndPredicateWithSubpredicates( CFArrayRef subpredicates ) = CompoundPredicateRef  
CompoundPredicateNotPredicateWithSubpredicate( PredicateRef predicate ) = CompoundPredicateRef  
CompoundPredicateOrPredicateWithSubpredicates( CFArrayRef subpredicates ) = CompoundPredicateRef  
CompoundPredicateWithType( NSCompoundPredicateType type, CFArrayRef subpredicates ) = CompoundPredicateRef
```

##### Info

```
CompoundPredicateType( CompoundPredicateRef ref ) = NSCompoundPredicateType  
CompoundPredicateSubpredicates( CompoundPredicateRef ref ) = CFArrayRef
```

#### Apple documentation

[NSCompoundPredicate](#)





## Control

### Description

Cocoa widgets that inherit from control (NSControl) can be manipulated with the following functions.

### Functions

```
ControlWithTag( NSInteger tag ) = CocoaControlRef
ControlExists( NSInteger tag ) = Boolean
```

#### Enabling and disabling

```
ControlIsEnabled( NSInteger tag ) = Boolean
ControlSetEnabled( NSInteger tag, Boolean flag )
```

#### Value

```
ControlDoubleValue( NSInteger tag ) = double
ControlSetDoubleValue( NSInteger tag, double value )
ControlIntegerValue( NSInteger tag ) = long
ControlSetIntegerValue( NSInteger tag, NSInteger value )
ControlStringValue( NSInteger tag ) = CFStringRef
ControlSetStringValue( NSInteger tag, CFStringRef string )
ControlSetAttributedStringValue( NSInteger tag, CFAttributedStringRef string )
```

#### Interacting with other controls

```
ControlTakeDoubleValueFrom( NSInteger tag, NSInteger fromViewTag )
ControlTakeIntegerValueFrom( NSInteger tag, NSInteger fromViewTag )
ControlTakeObjectValueFrom( NSInteger tag, NSInteger otherCtrlTag )
ControlTakeStringValueFrom( NSInteger tag, NSInteger fromViewTag )
```

#### Formatting text

```
ControlSetAlignment( NSInteger tag, NSTextAlignment alignment )
ControlFont( NSInteger tag ) = CTNSFontRef
ControlSetFont( NSInteger tag, CTNSFontRef font )
ControlSetFontWithName( NSInteger tag, CFStringRef name, CGFloat size )
ControlLineBreakMode( NSInteger tag ) = NSLineBreakMode
ControlSetLineBreakMode( NSInteger tag, NSLineBreakMode mode )
ControlUsesSingleLineMode( NSInteger tag ) = Boolean
ControlSetUsesSingleLineMode( NSInteger tag, Boolean flag )
```

#### Expansion tooltips

```
ControlDrawWithExpansionFrameInView( NSInteger ctrlTag, CGRect contentFrame, NSInteger vwTag )macOS 10.10+
ControlAllowsExpansionToolTips( NSInteger tag ) = BooleanmacOS 10.8+
ControlSetAllowsExpansionToolTips( NSInteger tag, Boolean flag )macOS 10.8+
```

#### Size

```
ControlSetSize( NSInteger tag, NSControlSize size )
ControlSizeThatFits( NSInteger tag, CGSize size ) = CGSizemacOS 10.10
ControlSizeToFit( NSInteger tag )
```

#### Target/Action

```
ControlSetAction( NSInteger tag, CFStringRef actionName )
ControlSetContinuous( NSInteger tag, Boolean flag )
ControlSendActionOn( NSInteger tag, NSEventMask mask )
```

#### Tags

```
ControlTag( CocoaControlRef ref ) = NSInteger
ControlSetTag( CocoaControlRef ref, NSInteger tag )
```

#### Activating from keyboard

```
ControlPerformClick( NSInteger tag )
ControlRefusesFirstResponder( NSInteger tag ) = Boolean
ControlSetRefusesFirstResponder( NSInteger tag, Boolean flag )
```

#### Tracking mouse

```
ControlIgnoresMultiClick( NSInteger tag ) = Boolean
ControlSetIgnoresMultiClick( NSInteger tag, Boolean flag )
```

#### Custom

```
ControlSetTargetAction( NSInteger ctrlTag, NSInteger targetTag, CFStringRef actionName )
ControlSetActionCallback( NSInteger tag, ptr callback, ptr userData )
ControlSetFormat( NSInteger tag, CFStringRef characters, Boolean pass, NSUInteger maxLength, NSUInteger caseOption )
```

## Apple documentation



---

## CountedSet

### Functions

Init

```
CountedSetWithArray( CFArrayRef array ) = CountedSetRef
CountedSetWithSet( CFSetRef set ) = CountedSetRef
CountedSetWithCapacity( NSUInteger length ) = CountedSetRef
```

Add/remove

```
CountedSetAddObject( CountedSetRef cs, CFTypeRef obj )
CountedSetRemoveObject( CountedSetRef cs, CFTypeRef obj )
```

Combine/recombine

```
CountedSetUnionSet( CountedSetRef cs, CFSetRef set )
CountedSetMinusSet( CountedSetRef cs, CFSetRef set )
CountedSetIntersectSet( CountedSetRef cs, CFSetRef set )
```

Examine

```
CountedSetCountForObject( CountedSetRef cs, CFTypeRef obj ) = NSUInteger
```

Accessing members

```
CountedSetObjectEnumerator( CountedSetRef cs ) = EnumeratorRef
```

### Apple documentation

[NSCountedSet](#)



---

## Cursor

### Description

Functions for manipulating the cursor.

### Functions

Init  
`CursorWithImage( ImageRef image, CGPoint hotSpot ) = CursorRef`

Attributes  
`CursorImage( CursorRef cursor ) = ImageRef`  
`CursorHotSpot( CursorRef cursor ) = CGPoint`  
`CursorHide`  
`CursorUnhide`  
`CursorSetHiddenUntilMouseMoves( Boolean flag )`

Controlling which cursor is current  
`CursorPop`  
`CursorPush( CursorRef cursor )`  
`CursorSet( CursorRef cursor )`

Cursor instances  
`CursorCurrent = CursorRef`  
`CursorCurrentSystem = CursorRef`  
`CursorArrow = CursorRef`  
`CursorContextMenu = CursorRef`  
`CursorClosedHand = CursorRef`  
`CursorCrosshair = CursorRef`  
`CursorDisappearingItem = CursorRef`  
`CursorDragCopy = CursorRef`  
`CursorDragLink = CursorRef`  
`CursorIBeam = CursorRef`  
`CursorOpenHand = CursorRef`  
`CursorOperationNotAllowed = CursorRef`  
`CursorPointingHand = CursorRef`  
`CursorResizeDown = CursorRef`  
`CursorResizeLeft = CursorRef`  
`CursorResizeRight = CursorRef`  
`CursorResizeUp = CursorRef`  
`CursorResizeUpDown = CursorRef`  
`CursorIBeamForVerticalLayout = CursorRef`

Convenience  
`CursorSetImage( ImageRef image, CGPoint hotSpot )`  
`CursorSetImageNamed( CFStringRef imageName, CGPoint hotSpot )`  
`CursorSetArrow`  
`CursorSetContextMenu`  
`CursorSetClosedHand`  
`CursorSetCrosshair`  
`CursorSetDisappearingItem`  
`CursorSetDragCopy`  
`CursorSetDragLink`  
`CursorSetIBeam`  
`CursorSetOpenHand`  
`CursorSetOperationNotAllowed`  
`CursorSetPointingHand`  
`CursorSetResizeDown`  
`CursorSetResizeLeft`  
`CursorSetResizeRight`  
`CursorSetResizeUp`  
`CursorSetResizeUpDown`  
`CursorSetIBeamForVerticalLayout`

### Apple documentation

[NSCursor](#)



---

## Data

### Functions

Create

```
DataWithBytes( ptr bytes, NSUInteger length ) = CFDataRef
```

Reading

```
DataWithContentsOfURL( CFURLRef url, NSDataReadingOptions options, ErrorRef *err ) = CFDataRef
```

Writing

```
DataWriteToURL( CFDataRef dta, CFURLRef url, NSDataWritingOptions options, ErrorRef *err ) = Boolean
```

Encoding/decoding Base64 representation

```
//DataBase64EncodedData( CFStringRef string, NSDataBase64EncodingOptions options ) = CFDataRefmacOS 10.9+
```

Accessing bytes

```
DataBytes( CFDataRef dta ) = ptr
```

```
DataGetBytes( CFDataRef dta, ptr buffer, NSUInteger length )
```

```
DataGetBytesInRange( CFDataRef dta, ptr buffer, CFRange range )
```

Finding

```
DataSubdata( CFDataRef dta, CFRange range ) = CFDataRef
```

```
DataRangeOfData( CFDataRef dta, CFDataRef dataToFind, NSDataSearchOptions searchOptions, CFRange searchRange ) =  
CFRange
```

Testing

```
DataIsEqualToData( CFDataRef dtal, CFDataRef dta2 ) = Boolean
```

```
DataLength( CFDataRef dta ) = NSUInteger
```

Describing

```
DataDescription( CFDataRef dta ) = CFStringRef
```

• Mutable data •

Create

```
DataWithCapacity( NSUInteger size ) = CFMutableDataRef
```

```
DataWithLength( NSUInteger length ) = CFMutableDataRef
```

Access

```
DataMutableBytes( CFMutableDataRef dta ) = ptr
```

Length

```
DataSetLength( CFMutableDataRef dta, NSUInteger length )
```

Add

```
DataAppendBytes( CFMutableDataRef dta, ptr bytes, NSUInteger length )
```

```
DataAppendData( CFMutableDataRef dtal, CFDataRef dta2 )
```

```
DataIncreaseLengthBy( CFMutableDataRef dta, NSUInteger extraLength )
```

Modify

```
DataReplaceBytesInRange( CFMutableDataRef dta, ptr replacementBytes, CFRange range, NSUInteger length )
```

```
DataResetBytesInRange( CFMutableDataRef dta, CFRange range )
```

```
DataSetData( CFMutableDataRef dtal, CFDataRef dta2 )
```

### Apple documentation

[NSData](#)

[NSMutableData](#)



---

## DataDetector

### Functions

Create

```
DataDetectorWithTypes( NSTextCheckingTypes types, ErrorRef *err ) = DataDetectorRef
```

Checking types

```
DataDetectorCheckingTypes( DataDetectorRef ref ) = NSTextCheckingTypes
```

### Apple documentation

[NSDataDetector](#)



---

## Date

### Functions

Create

```
DateInit = CFDateRef// autoreleased
DateWithTimeIntervalSinceNow( CFTimeInterval secs ) = CFDateRef
DateWithTimeIntervalSinceDate( CFTimeInterval secs, CFDateRef date ) = CFDateRef
DateWithTimeIntervalSinceReferenceDate( CFTimeInterval secs ) = CFDateRef
DateWithTimeIntervalSince1970( CFTimeInterval secs ) = CFDateRef
```

Temporal boundaries

```
DateDistantFuture = CFDateRef
DateDistantPast = CFDateRef
```

Comparing

```
DateIsEqualToDate( CFDateRef dt1, CFDateRef dt2 ) = Boolean
DateEarlierDate( CFDateRef dt1, CFDateRef dt2 ) = CFDateRef
DateLaterDate( CFDateRef dt1, CFDateRef dt2 ) = CFDateRef
DateCompare( CFDateRef dt1, CFDateRef dt2 ) = NSComparisonResult
```

Get time intervals

```
DateTimeIntervalSinceDate( CFDateRef dt1, CFDateRef dt2 ) = CFTimeInterval
DateTimeIntervalSinceNow( CFDateRef dt ) = CFTimeInterval
DateTimeIntervalSinceReferenceDate( CFDateRef dt ) = CFTimeInterval
DateTimeIntervalSince1970( CFDateRef dt ) = CFTimeInterval
```

Add time intervals

```
DateByAddingTimeInterval( CFDateRef dt, CFTimeInterval secs ) = CFDateRef
```

Description

```
DateDescription( CFDateRef dt ) = CFStringRef
DateDescriptionWithLocale( CFDateRef dt, CFLocaleRef locale ) = CFStringRef
```

Custom

```
DateStringWithFormat( CFStringRef format ) = CFStringRef
```

### Apple documentation

[NSDate](#)



---

## DateComponents

### Functions

Init

```
DateComponentsInit = DateComponentsRef// autoreleased
```

Calendar and time zone

```
DateComponentsCalendar( DateComponentsRef dc ) = CFCalendarRef
DateComponentsSetCalendar( DateComponentsRef dc, CFCalendarRef cal )
DateComponentsTimeZone( DateComponentsRef dc ) = CFTimeZoneRef
DateComponentsSetTimeZone( DateComponentsRef dc, CFTimeZoneRef tz )
```

Validate date

```
DateComponentsIsValidDate( DateComponentsRef dc ) = Boolean// macOS 10.9+
DateComponentsIsValidDateInCalendar( DateComponentsRef dc, CFCalendarRef cal ) = Boolean// macOS 10.9+
DateComponentsDate( DateComponentsRef dc ) = CFDateRef
```

Years and months

```
DateComponentsEra( DateComponentsRef dc ) = NSInteger
DateComponentsSetEra( DateComponentsRef dc, NSInteger era )
DateComponentsYear( DateComponentsRef dc ) = NSInteger
DateComponentsSetYear( DateComponentsRef dc, NSInteger year )
DateComponentsYearForWeekOfYear( DateComponentsRef dc ) = NSInteger
DateComponentsSetYearForWeekOfYear( DateComponentsRef dc, NSInteger year )
DateComponentsQuarter( DateComponentsRef dc ) = NSInteger
DateComponentsSetQuarter( DateComponentsRef dc, NSInteger quarter )
DateComponentsMonth( DateComponentsRef dc ) = NSInteger
DateComponentsSetMonth( DateComponentsRef dc, NSInteger month )
DateComponentsIsLeapMonth( DateComponentsRef dc ) = Boolean// macOS 10.8+
DateComponentsSetLeapMonth( DateComponentsRef dc, Boolean flag )// macOS 10.8+
```

Weeks and days

```
DateComponentsWeekday( DateComponentsRef dc ) = NSInteger
DateComponentsSetWeekday( DateComponentsRef dc, NSInteger day )
DateComponentsWeekdayOrdinal( DateComponentsRef dc ) = NSInteger
DateComponentsSetWeekdayOrdinal( DateComponentsRef dc, NSInteger day )
DateComponentsWeekOfMonth( DateComponentsRef dc ) = NSInteger
DateComponentsSetWeekOfMonth( DateComponentsRef dc, NSInteger week )
DateComponentsWeekOfYear( DateComponentsRef dc ) = NSInteger
DateComponentsSetWeekOfYear( DateComponentsRef dc, NSInteger week )
DateComponentsDay( DateComponentsRef dc ) = NSInteger
DateComponentsSetDay( DateComponentsRef dc, NSInteger day )
```

Hours and seconds

```
DateComponentsHour( DateComponentsRef dc ) = NSInteger
DateComponentsSetHour( DateComponentsRef dc, NSInteger hour )
DateComponentsMinute( DateComponentsRef dc ) = NSInteger
DateComponentsSetMinute( DateComponentsRef dc, NSInteger minute )
DateComponentsSecond( DateComponentsRef dc ) = NSInteger
DateComponentsSetSecond( DateComponentsRef dc, NSInteger second )
DateComponentsNanosecond( DateComponentsRef dc ) = NSInteger// macOS 10.9+
DateComponentsSetNanosecond( DateComponentsRef dc, NSInteger nanosecond )// macOS 10.9+
```

Components as calendrical units

```
DateComponentsValueForComponent( DateComponentsRef dc, NSCalendarUnit unit ) = NSInteger// macOS 10.9+
DateComponentsSetValueForComponent( DateComponentsRef dc, NSInteger value, NSCalendarUnit unit )// macOS 10.9+
```

### Apple documentation

[NSDateComponents](#)



## DateComponentsFormatter

### Functions

Formatting values

```
DateComponentsFormatterStringFromDateComponents( DateComponentsFormatterRef formatter, DateComponentsRef dc ) = CFStringRef
DateComponentsFormatterStringForObjectValue( DateComponentsFormatterRef formatter, CFTypeRef obj ) = CFStringRef
DateComponentsFormatterStringFromDate( DateComponentsFormatterRef formatter, CFDateRef startDate, CFDateRef endDate ) = CFStringRef
DateComponentsFormatterStringFromTimeInterval( DateComponentsFormatterRef formatter, CFTimeInterval interval ) = CFStringRef
DateComponentsFormatterLocalizedStringFromDateComponents( DateComponentsFormatterRef formatter, DateComponentsRef dc, NSDateComponentsFormatterUnitsStyle style ) = CFStringRef
```

Configure formatter options

```
DateComponentsFormatterAllowedUnits( DateComponentsFormatterRef formatter ) = NSCalendarUnit
DateComponentsFormatterSetAllowedUnits( DateComponentsFormatterRef formatter, NSCalendarUnit units )
DateComponentsFormatterAllowsFractionalUnits( DateComponentsFormatterRef formatter ) = Boolean
DateComponentsFormatterSetAllowsFractionalUnits( DateComponentsFormatterRef formatter, Boolean flag )
DateComponentsFormatterCalendar( DateComponentsFormatterRef formatter ) = CFCalendarRef
DateComponentsFormatterSetCalendar( DateComponentsFormatterRef formatter, CFCalendarRef cal )
DateComponentsFormatterCollapsesLargestUnit( DateComponentsFormatterRef formatter ) = Boolean
DateComponentsFormatterSetCollapsesLargestUnit( DateComponentsFormatterRef formatter, Boolean flag )
DateComponentsFormatterIncludesApproximationPhrase( DateComponentsFormatterRef formatter ) = Boolean
DateComponentsFormatterSetIncludesApproximationPhrase( DateComponentsFormatterRef formatter, Boolean flag )
DateComponentsFormatterIncludesTimeRemainingPhrase( DateComponentsFormatterRef formatter ) = Boolean
DateComponentsFormatterSetIncludesTimeRemainingPhrase( DateComponentsFormatterRef formatter, Boolean flag )
DateComponentsFormatterMaximumUnitCount( DateComponentsFormatterRef formatter ) = NSInteger
DateComponentsFormatterSetMaximumUnitCount( DateComponentsFormatterRef formatter, NSInteger count )
DateComponentsFormatterUnitsStyle( DateComponentsFormatterRef formatter ) = NSDateComponentsFormatterUnitsStyle
DateComponentsFormatterSetUnitsStyle( DateComponentsFormatterRef formatter, NSDateComponentsFormatterUnitsStyle style )
DateComponentsFormatterZeroFormattingBehavior( DateComponentsFormatterRef formatter ) = NSDateComponentsFormatterZeroFormattingBehavior
DateComponentsFormatterSetZeroFormattingBehavior( DateComponentsFormatterRef formatter, NSDateComponentsFormatterZeroFormattingBehavior behavior )
```

Instance properties

```
DateComponentsFormatterFormattingContext( DateComponentsFormatterRef formatter ) = NSFormattingContext
DateComponentsFormatterSetFormattingContext( DateComponentsFormatterRef formatter, NSFormattingContext ctx )
DateComponentsFormatterReferenceDate( DateComponentsFormatterRef formatter ) = CFDateRef // macOS 10.13+
DateComponentsFormatterSetReferenceDate( DateComponentsFormatterRef formatter, CFDateRef dt ) // macOS 10.13+
```

Instance methods

```
DateComponentsFormatterGetObjectValue( DateComponentsFormatterRef formatter, CFTypeRef *obj, CFStringRef string, CFStringRef *errorDescription ) = Boolean
```

### Apple documentation

[NSDateComponentsFormatter](#)





## DateFormatter

### Functions

Init  
`DateFormatterInit = DateFormatterRef// autoreleased`

#### Converting objects

```
DateFormatterDateFromString( DateFormatterRef formatter, CFStringRef string ) = CFDateRef
DateFormatterStringFromDate( DateFormatterRef formatter, CFDateRef dt ) = CFStringRef
DateFormatterLocalizedStringFromDate( CFDateRef dt, NSDateFormatterStyle dateStyle, NSDateFormatterStyle timeStyle ) = CFStringRef
DateFormatterGetObjectValueForString( DateFormatterRef formatter, CTypeRef *obj, CFStringRef string, CFRange *inOutRange, ErrorRef *err ) = Boolean
```

#### Format and style

```
DateFormatterDateStyle( DateFormatterRef formatter ) = NSDateFormatterStyle
DateFormatterSetDateStyle( DateFormatterRef formatter, NSDateFormatterStyle style )
DateFormatterTimeStyle( DateFormatterRef formatter ) = NSDateFormatterStyle
DateFormatterSetTimeStyle( DateFormatterRef formatter, NSDateFormatterStyle style )
DateFormatterDateFormat( DateFormatterRef formatter ) = CFStringRef
DateFormatterSetDateFormat( DateFormatterRef formatter, CFStringRef format )
DateFormatterSetLocalizedDateFormatFromTemplate( DateFormatterRef formatter, CFStringRef template )// macOS 10.10+
DateFormatterDateFormatFromTemplate( DateFormatterRef formatter, NSUInteger options, CFLocaleRef locale ) = CFStringRef
DateFormatterFormattingContext( DateFormatterRef formatter ) = NSFormattingContext// macOS 10.10+
DateFormatterSetFormattingContext( DateFormatterRef formatter, NSFormattingContext ctx )// macOS 10.10+
```

#### Attributes

```
DateFormatterCalendar( DateFormatterRef formatter ) = CFCalendarRef
DateFormatterSetCalendar( DateFormatterRef formatter, CFCalendarRef cal )
DateFormatterDefaultDate( DateFormatterRef formatter ) = CFDateRef
DateFormatterSetDefaultDate( DateFormatterRef formatter, CFDateRef dt )
DateFormatterLocale( DateFormatterRef formatter ) = CFLocaleRef
DateFormatterSetLocale( DateFormatterRef formatter, CFLocaleRef locale )
DateFormatterTimeZone( DateFormatterRef formatter ) = CFTimeZoneRef
DateFormatterSetTimeZone( DateFormatterRef formatter, CFTimeZoneRef zone )
DateFormatterTwoDigitStartDate( DateFormatterRef formatter ) = CFDateRef
DateFormatterSetTwoDigitStartDate( DateFormatterRef formatter, CFDateRef dt )
DateFormatterGregorianStartDate( DateFormatterRef formatter ) = CFDateRef
DateFormatterSetGregorianStartDate( DateFormatterRef formatter, CFDateRef dt )
```

#### Behavior

```
DateFormatterBehavior( DateFormatterRef formatter ) = NSDateFormatterBehavior
DateFormatterSetBehavior( DateFormatterRef formatter, NSDateFormatterBehavior behavior )
DateFormatterDefaultBehavior = NSDateFormatterBehavior
DateFormatterSetDefaultBehavior( NSDateFormatterBehavior behavior )
```

#### Natural language

```
DateFormatterIsLenient( DateFormatterRef formatter ) = Boolean
DateFormatterSetLenient( DateFormatterRef formatter, Boolean flag )
DateFormatterDoesRelativeDateFormatting( DateFormatterRef formatter ) = Boolean
DateFormatterSetDoesRelativeDateFormatting( DateFormatterRef formatter, Boolean flag )
```

#### AM and PM symbols

```
DateFormatterAMSymbol( DateFormatterRef formatter ) = CFStringRef
DateFormatterSetAMSymbol( DateFormatterRef formatter, CFStringRef symbol )
DateFormatterPMSymbol( DateFormatterRef formatter ) = CFStringRef
DateFormatterSetPMSymbol( DateFormatterRef formatter, CFStringRef symbol )
```

#### Weekday symbols

```
DateFormatterWeekdaySymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetWeekdaySymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterShortWeekdaySymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetShortWeekdaySymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterVeryShortWeekdaySymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetVeryShortWeekdaySymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterStandaloneWeekdaySymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetStandaloneWeekdaySymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterShortStandaloneWeekdaySymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetShortStandaloneWeekdaySymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterVeryShortStandaloneWeekdaySymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetVeryShortStandaloneWeekdaySymbols( DateFormatterRef formatter, CFArrayRef symbols )
```

#### Month symbols

```
DateFormatterMonthSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetMonthSymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterShortMonthSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetShortMonthSymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterVeryShortMonthSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetVeryShortMonthSymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterStandaloneMonthSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetStandaloneMonthSymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterShortStandaloneMonthSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetShortStandaloneMonthSymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterVeryShortStandaloneMonthSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetVeryShortStandaloneMonthSymbols( DateFormatterRef formatter, CFArrayRef symbols )
```

#### Quarter symbols

```
DateFormatterQuarterSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetQuarterSymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterShortQuarterSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetShortQuarterSymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterStandaloneQuarterSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetStandaloneQuarterSymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterShortStandaloneQuarterSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetShortStandaloneQuarterSymbols( DateFormatterRef formatter, CFArrayRef symbols )
```

#### Era symbols

```
DateFormatterEraSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetEraSymbols( DateFormatterRef formatter, CFArrayRef symbols )
DateFormatterLongEraSymbols( DateFormatterRef formatter ) = CFArrayRef
DateFormatterSetLongEraSymbols( DateFormatterRef formatter, CFArrayRef symbols )
```

### Apple documentation

[NSDateFormatter](#)



---

## DateInterval

### Functions

Create

```
DateIntervalInit = DateIntervalRef// autoreleased  
DateIntervalWithStartDateDuration( CFDateRef dt, CFTimeInterval ti ) = DateIntervalRef  
DateIntervalWithStartDateEndDate( CFDateRef startDate, CFDateRef endDate ) = DateIntervalRef
```

Start/end date and duration

```
DateIntervalStartDate( DateIntervalRef di ) = CFDateRef  
DateIntervalEndDate( DateIntervalRef di ) = CFDateRef  
DateIntervalDuration( DateIntervalRef di ) = CFTimeInterval
```

Compare date intervals

```
DateIntervalCompare( DateIntervalRef di1, DateIntervalRef di2 ) = NSComparisonResult  
DateIntervalIsEqualTo( DateIntervalRef di1, DateIntervalRef di2 ) = Boolean
```

Determine intersections

```
DateIntervalIntersectsDateInterval( DateIntervalRef di1, DateIntervalRef di2 ) = Boolean  
DateIntervalIntersectionWithDateInterval( DateIntervalRef di1, DateIntervalRef di2 ) = DateIntervalRef
```

Determine if date occurs in date interval

```
DateIntervalContainsDate( DateIntervalRef di, CFDateRef dt ) = Boolean
```

### Apple documentation

[NSDateInterval](#)



---

## DateIntervalFormatter

### Functions

#### Init

```
DateIntervalFormatterInit = DateIntervalFormatterRef// autoreleased
```

#### Formatting

```
DateIntervalFormatterStringFromDate( DateIntervalFormatterRef dif, CFDateRef fromDate, CFDateRef toDate ) = CFStringRef
```

#### Configure

```
DateIntervalFormatterDateStyle( DateIntervalFormatterRef dif ) = NSDateIntervalFormatterStyle
DateIntervalFormatterSetDateStyle( DateIntervalFormatterRef dif, NSDateIntervalFormatterStyle style )
DateIntervalFormatterTimeStyle( DateIntervalFormatterRef dif ) = NSDateIntervalFormatterStyle
DateIntervalFormatterSetTimeStyle( DateIntervalFormatterRef dif, NSDateIntervalFormatterStyle style )
DateIntervalFormatterDateTemplate( DateIntervalFormatterRef dif ) = CFStringRef
DateIntervalFormatterSetDateTemplate( DateIntervalFormatterRef dif, CFStringRef template )
DateIntervalFormatterCalendar( DateIntervalFormatterRef dif ) = CFCalendarRef
DateIntervalFormatterSetCalendar( DateIntervalFormatterRef dif, CFCalendarRef cal )
DateIntervalFormatterLocale( DateIntervalFormatterRef dif ) = CFLocaleRef
DateIntervalFormatterSetLocale( DateIntervalFormatterRef dif, CFLocaleRef locale )
DateIntervalFormatterTimeZone( DateIntervalFormatterRef dif ) = CFTimeZoneRef
DateIntervalFormatterSetTimeZone( DateIntervalFormatterRef dif, CFTimeZoneRef tz )
```

#### Instance

```
DateIntervalFormatterStringFromDateInterval( DateIntervalFormatterRef dif, DateIntervalRef di ) = CFStringRef
```

### Apple documentation

[NSDateIntervalFormatter](#)



## DatePicker

statement

### Syntax

**datepicker** *tag, enabled, date, rect, style, elements, wndTag*

### Description

The **datepicker** statement puts a new datepicker in the current output cocoa window, or alters an existing datepicker's characteristics.

### Parameters

<i>tag</i>	A number to identify the datepicker. A negative tag hides the datepicker
<i>enabled</i>	Enables or disables the datepicker
<i>date</i>	The date to be displayed in the datepicker.
<i>rect</i>	Origin and size of the datepicker in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>style</i>	The datepicker style. <a href="#">NSTextFieldAndStepperDatePickerStyle</a> (default) <a href="#">NSClockAndCalendarDatePickerStyle</a> <a href="#">NSTextFieldDatePickerStyle</a>
<i>elements</i>	The elements displayed in the datepicker. Can be combined using '+'. <a href="#">NSHourMinuteDatePickerElementFlag</a> (default) <a href="#">NSHourMinuteSecondDatePickerElementFlag</a> <a href="#">NSTimeZoneDatePickerElementFlag</a> <a href="#">NSYearMonthDatePickerElementFlag</a> <a href="#">NSYearMonthDayDatePickerElementFlag</a> <a href="#">NSEraDatePickerElementFlag</a>
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

[\\_btnClick](#)

### Functions

```
DatePickerWithTag( NSInteger tag ) = DatePickerRef
DatePickerExists( NSInteger tag ) = Boolean
```

Init

```
DatePickerInit( NSInteger tag, CGRect r ) = DatePickerRef// autoreleased
```

Configure

```
DatePickerIsBezeled( NSInteger tag ) = Boolean
DatePickerSetBezeled( NSInteger tag, Boolean flag )
DatePickerIsBordered( NSInteger tag ) = Boolean
DatePickerSetBordered( NSInteger tag, Boolean flag )
DatePickerBackgroundColor( NSInteger tag ) = ColorRef
DatePickerSetBackgroundColor( NSInteger tag, ColorRef col )
DatePickerDrawsBackground( NSInteger tag ) = Boolean
DatePickerSetDrawsBackground( NSInteger tag, Boolean flag )
DatePickerTextColor( NSInteger tag ) = ColorRef
DatePickerSetTextColor( NSInteger tag, ColorRef col )
DatePickerStyle( NSInteger tag ) = NSDatePickerStyle
DatePickerSetStyle( NSInteger tag, NSDatePickerStyle style )
DatePickerElements( NSInteger tag ) = NSDatePickerElementFlags
DatePickerSetElements( NSInteger tag, NSDatePickerElementFlags elements )
```

Range and mode

```
DatePickerCalendar( NSInteger tag ) = CFCalendarRef
DatePickerSetCalendar( NSInteger tag, CFCalendarRef cal )
DatePickerLocale( NSInteger tag ) = CFLocaleRef
DatePickerSetLocale( NSInteger tag, CFLocaleRef locale )
```

```
DatePickerMode( NSInteger tag ) = NSDatePickerMode
DatePickerSetMode( NSInteger tag, NSInteger mode )
DatePickerTimeZone( NSInteger tag ) = CFTimeZoneRef
DatePickerSetTimeZone( NSInteger tag, CFTimeZoneRef zone )
```

#### Values

```
DatePickerDateValue( NSInteger tag ) = CFDateRef
DatePickerSetDateValue( NSInteger tag, CFDateRef dt )
DatePickerTimeInterval( NSInteger tag ) = CFTimeInterval
DatePickerSetTimeInterval( NSInteger tag, CFTimeInterval ti )
```

#### Constraining

```
DatePickerMinDate( NSInteger tag ) = CFDateRef
DatePickerSetMinDate( NSInteger tag, CFDateRef dt )
DatePickerMaxDate( NSInteger tag ) = CFDateRef
DatePickerSetMaxDate( NSInteger tag, CFDateRef dt )
```

### See Also

[Control](#), [View](#)

### Apple documentation

[NSDatePicker](#)



## DecimalNumber

---

### Functions

#### Create

```
DecimalNumberWithDecimal( NSDecimal dcm ) = DecimalNumberRef
DecimalNumberWithMantissa( UInt64 mantissa, short exponent, Boolean isNegative ) = DecimalNumberRef
DecimalNumberWithString( CFStringRef string ) = DecimalNumberRef
DecimalNumberWithStringAndLocale( CFStringRef string, CFLocaleRef locale ) = DecimalNumberRef
DecimalNumberOne = DecimalNumberRef
DecimalNumberZero = DecimalNumberRef
DecimalNumberNotANumber = DecimalNumberRef
```

#### Perform arithmetic

```
DecimalNumberByAdding( DecimalNumberRef dn1, DecimalNumberRef dn2 ) = DecimalNumberRef
DecimalNumberBySubtracting( DecimalNumberRef dn1, DecimalNumberRef dn2 ) = DecimalNumberRef
DecimalNumberByMultiplyingBy( DecimalNumberRef dn1, DecimalNumberRef dn2 ) = DecimalNumberRef
DecimalNumberByDividingBy( DecimalNumberRef dn1, DecimalNumberRef dn2 ) = DecimalNumberRef
DecimalNumberByRaisingToPower( DecimalNumberRef dn, NSInteger power ) = DecimalNumberRef
DecimalNumberByMultiplyingByPowerOf10( DecimalNumberRef dn, short power ) = DecimalNumberRef
DecimalNumberByAddingWithBehavior( DecimalNumberRef dn1, DecimalNumberRef dn2, DecimalNumberBehaviorsRef behavior ) = DecimalNumberRef
DecimalNumberBySubtractingWithBehavior( DecimalNumberRef dn1, DecimalNumberRef dn2, DecimalNumberBehaviorsRef behavior ) = DecimalNumberRef
DecimalNumberByMultiplyingByWithBehavior( DecimalNumberRef dn1, DecimalNumberRef dn2, DecimalNumberBehaviorsRef behavior ) = DecimalNumberRef
DecimalNumberByDividingByWithBehavior( DecimalNumberRef dn1, DecimalNumberRef dn2, DecimalNumberBehaviorsRef behavior ) = DecimalNumberRef
DecimalNumberByRaisingToPowerWithBehavior( DecimalNumberRef dn, NSInteger power, DecimalNumberBehaviorsRef behavior ) = DecimalNumberRef
DecimalNumberByMultiplyingByPowerOf10WithBehavior( DecimalNumberRef dn, short power, DecimalNumberBehaviorsRef behavior ) = DecimalNumberRef
```

#### Rounding

```
DecimalNumberByRounding( DecimalNumberRef dn, DecimalNumberBehaviorsRef behavior ) = DecimalNumberRef
```

#### Behavior

```
DecimalNumberDefaultBehavior = DecimalNumberBehaviorsRef
```

#### Value

```
DecimalNumberDecimalValue( DecimalNumberRef dn ) = NSDecimal
DecimalNumberDoubleValue( DecimalNumberRef dn ) = double
DecimalNumberDescription( DecimalNumberRef dn, CFLocaleRef locale ) = CFStringRef
```

#### Compare

```
DecimalNumberCompare( DecimalNumberRef dn1, DecimalNumberRef dn2 ) = NSComparisonResult
```

#### Min/max values

```
DecimalNumberMaximum = DecimalNumberRef
DecimalNumberMinimum = DecimalNumberRef
```

### Apple documentation

[NSDecimalNumber](#)



---

## DecimalNumberBehaviors

### Functions

Rounding

```
DecimalNumberBehaviorsRoundingMode( DecimalNumberBehaviorsRef ref ) = NSRoundingMode
```

```
DecimalNumberBehaviorsScale( DecimalNumberBehaviorsRef ref ) = short
```

### Apple documentation

[NSDecimalNumberBehaviors](#)





---

### DecimalNumberHandler

#### Functions

Create

```
DecimalNumberHandlerDefault = DecimalNumberHandlerRef
```

```
DecimalNumberHandlerWithRoundingMode( NSRoundingMode mode, short scale, Boolean raiseOnExactness, Boolean  
raiseOnOverflow, Boolean raiseOnUnderflow, Boolean raiseOnDivideByZero ) = DecimalNumberHandlerRef
```

#### Apple documentation

[NSDecimalNumberHandler](#)



## Dialog

### Syntax

```
ev = dialog(0)
id = dialog(ev)
wnd = dialog(-1)
```

### Description

This supplements FBHelp's **dialog** function documentation.

With the exception of `_btnClick`, CocoaUI elements have their own dialog event constants.

Event Type	Description	ID
<code>_btnClick</code>	User clicked a button.	Button tag
<code>_comboBoxXxxxx</code>	See <a href="#">ComboBox</a> statement for event constants.	ComboBox tag
<code>_searchFieldXxxxx</code>	See <a href="#">SearchField</a> statement for event constants.	Field tag
<code>_splitViewXxxxx</code>	See <a href="#">SplitView</a> statement for event constants.	SplitView tag
<code>_tableViewXxxxx</code>	See <a href="#">TableView</a> statement for event constants.	TableView tag
<code>_tabViewXxxxx</code>	See <a href="#">TabView</a> statement for event constants.	TabView tag
<code>_textFieldXxxxx</code>	See <a href="#">TextField</a> statement for event constants.	Field tag
<code>_tokenFieldXxxxx</code>	See <a href="#">TokenField</a> statement for event constants.	Field tag
<code>_toolbarItemXxxxx</code>	See <a href="#">ToolbarItem</a> statement for event constants.	ToolbarItem tag
<code>_viewXxxxx</code>	See <a href="#">View</a> statement for event constants.	View tag
<code>_windowXxxxx</code>	See <a href="#">Cocoa Window</a> statement for event constants.	Window tag

### Functions

```
DialogEventBool = Boolean
DialogEventSetBool( Boolean value )
DialogEventLong = long
DialogEventSetLong( long value )
DialogEventFloat = float
DialogEventSetFloat( float value )
DialogEventDouble = double
DialogEventSetDouble( double value )
DialogEventSize = CGSize
DialogEventSetSize( CGSize value )
DialogEventRect = CGRect
DialogEventSetRect( CGRect value )
DialogEventViewTag = NSInteger
DialogEventSetViewTag( NSInteger value )
DialogEventArray = CFArrayRef
DialogEventURL = CFURLRef
DialogEventDraggingInfo = DraggingInfoRef
DialogEventSetDraggingInfo( DraggingInfoRef info )
DialogEventString = CFStringRef
DialogEventSetString( CFStringRef string )
DialogEventFont = FontRef
DialogEventColor = ColorRef
DialogEventSetColor( ColorRef col )
DialogEventObject = CTypeRef
DialogEventSetObject( CTypeRef obj )
DialogEventSetMenu( CocoaMenuRef menu )
```

### See Also

FBHelp [dialog](#) function, FBHelp on [dialog](#) statement  
[Event](#)



## Dictionary

### Functions

#### Create

```
DictionaryWithContentsOfURL( CFURLRef url, ErrorRef *err ) = CFDictionaryRef
DictionaryWithDictionary( CFDictionaryRef dict ) = CFDictionaryRef
DictionaryWithObject( CFTyperef obj, CFStringRef key ) = CFDictionaryRef
DictionaryWithObjects( CFTyperef obj, ... ) = CFDictionaryRef // a comma-separated list of objects and keys,
ending with NULL
DictionaryWithObjectsForKeys( CFDictionaryRef dict, CFArrayRef objects, CFArrayRef keys ) = CFDictionaryRef
```

#### Count

```
DictionaryCount( CFDictionaryRef dict ) = NSUInteger
```

#### Compare

```
DictionaryIsEqual( CFDictionaryRef dict, CFDictionary otherDict ) = Boolean
```

#### Accessing keys and values

```
DictionaryAllKeys( CFDictionaryRef dict ) = CFArrayRef
DictionaryAllKeysForObject( CFDictionaryRef dict, CFTyperef obj ) = CFArrayRef
DictionaryAllValues( CFDictionaryRef dict ) = CFArrayRef
DictionaryObjectForKey( CFDictionaryRef dict, CFStringRef key ) = CFTyperef
DictionaryObjectsForKeys( CFDictionaryRef dict, CFArrayRef keys, CFTyperef notFoundMarker ) = CFArrayRef
```

#### Enumerator

```
DictionaryKeyEnumerator( CFDictionaryRef dict ) = EnumeratorRef
DictionaryObjectEnumerator( CFDictionaryRef dict ) = EnumeratorRef
DictionaryEnumerateKeysAndObjects( CFDictionaryRef dict, ptr callback )
DictionaryEnumerateKeysAndObjectsWithOptions( CFDictionaryRef dict, NSEnumerationOptions options, ptr callback )
```

#### Sorting

```
DictionaryKeysSortedByValueUsingSelector( CFDictionaryRef dict, CFStringRef selector ) = CFArrayRef
DictionaryKeysSortedByValueUsingComparator( CFDictionaryRef dict, ptr comparatorFn, ptr userData ) = CFArrayRef
DictionaryKeysSortedByValueWithOptionsUsingComparator( CFDictionaryRef dict, NSSortOptions options, ptr
comparatorFn, ptr userData ) = CFArrayRef
```

#### Filtering

```
DictionaryKeysOfEntriesPassingTest( CFDictionaryRef dict, ptr testFn, ptr userData ) = CFSetRef
DictionaryKeysWithOptionsOfEntriesPassingTest( CFDictionaryRef dict, NSEnumerationOptions options, ptr testFn, ptr
userData ) = CFSetRef
```

#### Storing

```
DictionaryWriteToURL( CFDictionaryRef dict, CFURLRef url, ErrorRef *err ) = Boolean
```

#### File attributes

```
DictionaryFileSize( CFDictionaryRef dict ) = UInt64
DictionaryFileType( CFDictionaryRef dict ) = CFStringRef
DictionaryFileCreationDate( CFDictionaryRef dict ) = CFDateRef
DictionaryFileModificationDate( CFDictionaryRef dict ) = CFDateRef
DictionaryFilePosixPermissions( CFDictionaryRef dict ) = NSUInteger
DictionaryFileOwnerAccountID( CFDictionaryRef dict ) = CFNumberRef
DictionaryFileOwnerAccountName( CFDictionaryRef dict ) = CFStringRef
DictionaryFileGroupOwnerAccountID( CFDictionaryRef dict ) = CFNumberRef
DictionaryFileGroupOwnerAccountName( CFDictionaryRef dict ) = CFStringRef
DictionaryFileExtensionHidden( CFDictionaryRef dict ) = Boolean
DictionaryFileIsImmutable( CFDictionaryRef dict ) = Boolean
DictionaryFileIsAppendOnly( CFDictionaryRef dict ) = Boolean
DictionaryFileSystemFileNumber( CFDictionaryRef dict ) = NSUInteger
DictionaryFileHFSTypeCode( CFDictionaryRef dict ) = OSType
DictionaryFileHFSCreatorCode( CFDictionaryRef dict ) = OSType
```

#### Description

```
DictionaryDescription( CFDictionaryRef dict ) = CFStringRef
DictionaryDescriptionInStringsFileFormat( CFDictionaryRef dict ) = CFStringRef
DictionaryDescriptionWithLocale( CFDictionaryRef dict, CFLocaleRef locale ) = CFStringRef
DictionaryDescriptionWithLocaleIndent( CFDictionaryRef dict, CFLocaleRef locale, NSUInteger indent ) = CFStringRef
```

#### • Mutable dictionary •

```
DictionaryWithCapacity( NSUInteger numItems ) = CFMutableDictionaryRef
DictionarySetObject( CFMutableDictionaryRef dict, CFTyperef obj, CFStringRef key )
DictionaryAddEntriesFromDictionary( CFMutableDictionaryRef dict, CFDictionaryRef otherDict )
DictionarySetDictionary( CFMutableDictionaryRef dict, CFDictionaryRef otherDict )
```

```
DictionaryRemoveObjectForKey( CFMutableDictionaryRef dict, CFStringRef key )  
DictionaryRemoveAllObjects( CFMutableDictionaryRef dict )  
DictionaryRemoveObjectsForKeys( CFMutableDictionaryRef dict, CFArrayRef keys )
```

## **Apple documentation**

[NSDictionary](#)

[NSMutableDictionary](#)



---

### DirectoryEnumerator

#### Functions

Attributes

```
DirectoryEnumeratorDirectoryAttributes( DirectoryEnumeratorRef enumerator ) = CFDictionaryRef  
DirectoryEnumeratorFileAttributes( DirectoryEnumeratorRef enumerator ) = CFDictionaryRef  
DirectoryEnumeratorLevel( DirectoryEnumeratorRef enumerator ) = NSUInteger
```

Skipping

```
DirectoryEnumeratorSkipDescendants( DirectoryEnumeratorRef enumerator )
```

#### Apple documentation

[NSDirectoryEnumerator](#)



---

## DistributedNotificationCenter

### Functions

Default center

```
DistributedNotificationCenterDefaultCenter = DistributedNotificationCenterRef
```

```
DistributedNotificationCenterForType( NSDistributedNotificationCenterType type ) = DistributedNotificationCenterRef
```

Observers

```
DistributedNotificationCenterAddObserver( ptr callback, NSNotificationName name, CTypeRef obj,  
NSNotificationSuspensionBehavior behavior )
```

```
DistributedNotificationCenterRemoveObserver( ptr callback, NSNotificationName name )
```

Notifications

```
DistributedNotificationCenterPostNotificationName( NSNotificationName name, CTypeRef obj, CFDictionaryRef userInfo,  
NSUInteger options )
```

Suspend

```
DistributedNotificationCenterSuspended( DistributedNotificationCenterRef ref ) = Boolean
```

```
DistributedNotificationCenterSetSuspended( DistributedNotificationCenterRef ref, Boolean flag )
```

### Apple documentation

[NSDistributedNotificationCenter](#)



---

## DockTile

### Functions

Content

```
DockTileContentView( DockTileRef tile ) = ViewRef
```

Info

```
DockTileSize( DockTileRef tile ) = CGSize
```

Badge icons

```
DockTileShowsApplicationBadge( DockTileRef tile ) = Boolean  
DockTileSetShowsApplicationBadge( DockTileRef tile, Boolean flag )  
DockTileBadgeLabel( DockTileRef tile ) = CFStringRef  
DockTileSetBadgeLabel( DockTileRef tile, CFStringRef label )
```

Updating

```
DockTileDisplay( DockTileRef tile )
```

### Apple documentation

[NSDockTile](#)



---

## DraggingInfo

### Functions

Dragging session info

```
DraggingInfoPasteboard( DrggingInfoRef info ) = CocoaPasteboardRef
DraggingInfoSequenceNumber( DraggingInfoRef info ) = NSInteger
DraggingInfoSource( DraggingInfoRef info ) = CTypeRef
DraggingInfoSourceOperationMask( DraggingInfoRef info ) = NSDragOperation
DraggingInfoLocation( DraggingInfoRef info ) = CGPoint
DraggingInfoDestinationWindow( DraggingInfoRef info ) = NSInteger
DraggingInfoNumberOfValidItemsForDrop( DraggingInfoRef info ) = NSInteger
```

Image info

```
DraggingInfoImageLocation( DraggingInfoRef info ) = CGPoint
```

// Sliding the image

```
DraggingInfoSlideImageTo( DraggingInfoRef info, CGPoint screenPt )
DraggingInfoAnimatesToDestination( DraggingInfoRef info ) = Boolean
DraggingInfoSetAnimatesToDestination( DraggingInfoRef info, Boolean flag )
DraggingInfoFormation( DraggingInfoRef info ) = NSDraggingFormation
```

Enumerate

```
DraggingInfoEnumerateItems( DraggingInfoRef info, NSDraggingItemEnumerationOptions options, NSInteger forViewTag,
CFArrayRef classes, CFDictionaryRef searchOptions, ptr callback, ptr userData )
```

Spring-loading support

```
DraggingInfoSpringLoadingHighlight( DraggingInfoRef info ) = NSSpringLoadingHighlight// macOS 10.11+
DraggingInfoResetSpringLoading( DraggingInfoRef info )// macOS 10.11+
```

### Apple documentation

[NSDraggingInfo](#)





EditMenu

statement

**Syntax**  
`editmenu menuIndex`

**Description**  
Builds a default edit menu. Requires the Cocoa runtime ([CocoaInit](#)).

Edit	
Undo	⌘Z
Redo	⇧⌘Z
Cut	⌘X
Copy	⌘C
Paste	⌘V
Delete	
Select All	⌘A



---

### Enumerator

#### Functions

Objects

```
EnumeratorAllObjects( EnumeratorRef enumerator ) = CFArrayRef
```

```
EnumeratorNextObject( EnumeratorRef enumerator ) = CTypeRef
```

#### Apple documentation

[NSEnumerator](#)



---

## Error

### Functions

Properties

```
ErrorCode( ErrorRef err ) = NSInteger
```

```
ErrorDomain( ErrorRef err ) = CFStringRef
```

```
ErrorUserInfo( ErrorRef err ) = CFDictionaryRef
```

Localized description

```
ErrorLocalizedDescription( ErrorRef err ) = CFStringRef
```

```
ErrorLocalizedRecoveryOptions( ErrorRef err ) = CFArrayRef
```

```
ErrorLocalizedRecoverySuggestion( ErrorRef err ) = CFStringRef
```

```
ErrorLocalizedFailureReason( ErrorRef err ) = CFStringRef
```

Display helpAnchor

```
ErrorHelpAnchor( ErrorRef err ) = CFStringRef
```

### Apple documentation

[NSError](#)



---

## Event

### Description

General events which are normally picked up in the on dialog xxxxx function.

### Functions

General info

```
EventLocationInWindow = CGPoint
EventModifierFlags = NSEventModifierFlags
EventTimestamp = CFTimeInterval
EventType = NSEventType
EventWindow = ptr
```

Key info

```
EventKeyRepeatDelay = CFTimeInterval
EventKeyRepeatInterval = CFTimeInterval
EventCharacters = CFStringRef
EventCharactersIgnoringModifiers = CFStringRef
EventIsARepeat = Boolean
EventKeyCode = unsigned short
```

Mouse info

```
EventPressedMouseButtons = NSUInteger
EventDoubleClickInterval = CFTimeInterval
EventMouseLocation = CGPoint
EventClickCount = NSInteger
```

Convenience

```
EventLocationInView( NSInteger tag ) = CGPoint
```

### Apple documentation

[NSEvent](#)



## Expression

---

### Functions

#### Init

```
ExpressionWithType( NSExpressionType type ) = ExpressionRef
ExpressionWithFormat( CFStringRef format, ... ) = ExpressionRef
ExpressionWithFormatAndArgumentArray( CFStringRef format, CFArrayRef argArray ) = ExpressionRef
ExpressionWithFormatAndArguments( CFStringRef format, va_list argList ) = ExpressionRef
```

#### Create for value

```
ExpressionForConstantValue( CTypeRef obj ) = ExpressionRef
ExpressionForEvaluatedObject = ExpressionRef
ExpressionForKeyPath( CFStringRef keyPath ) = ExpressionRef
ExpressionForVariable( CFStringRef string ) = ExpressionRef
ExpressionForAnyKey = ExpressionRef// macOS 10.9+
```

#### Create collection

```
ExpressionForAggregate( CFArrayRef subexpressions ) = ExpressionRef
ExpressionForUnionSet( ExpressionRef left, ExpressionRef right ) = ExpressionRef
ExpressionForIntersectSet( ExpressionRef left, ExpressionRef right ) = ExpressionRef
ExpressionForMinusSet( ExpressionRef left, ExpressionRef right ) = ExpressionRef
```

#### Create subquery

```
ExpressionForSubquery( ExpressionRef exp, CFStringRef variable, PredicateRef predicate ) = ExpressionRef
```

#### Create for function

```
ExpressionForFunction( CFStringRef name, CFArrayRef arguments ) = ExpressionRef
```

#### Info

```
ExpressionArguments( ExpressionRef ref ) = CFArrayRef
ExpressionCollection( ExpressionRef ref ) = CTypeRef
ExpressionConstantValue( ExpressionRef ref ) = CTypeRef
ExpressionType( ExpressionRef ref ) = NSExpressionType
ExpressionKeyPath( ExpressionRef ref ) = CFStringRef
ExpressionOperand( ExpressionRef ref ) = ExpressionRef
ExpressionPredicate( ExpressionRef ref ) = PredicateRef
ExpressionLeftExpression( ExpressionRef ref ) = ExpressionRef
ExpressionRightExpression( ExpressionRef ref ) = ExpressionRef
ExpressionVariable( ExpressionRef ref ) = CFStringRef
```

#### Evaluate

```
ExpressionValueWithObject( ExpressionRef exp, CTypeRef obj, CFMutableDictionaryRef context ) = CTypeRef
ExpressionAllowEvaluation( ExpressionRef ref )// macOS 10.9+
```

#### Initializers

```
ExpressionForConditional( PredicateRef predicate, ExpressionRef trueExpression, ExpressionRef falseExpression ) =
ExpressionRef// macOS 10.11+
```

#### Instance properties

```
ExpressionFalseExpression( ExpressionRef ref ) = ExpressionRef// macOS 10.11+
ExpressionTrueExpression( ExpressionRef ref ) = ExpressionRef// macOS 10.11+
```

### Apple documentation

[NSExpression](#)



## FileHandle

---

### Functions

#### Get

```
FileHandleForReadingFromURL( CFURLRef url, ErrorRef *err ) = FileHandleRef
FileHandleForWritingToURL( CFURLRef url, ErrorRef *err ) = FileHandleRef
FileHandleForUpdatingURL( CFURLRef url, ErrorRef *err ) = FileHandleRef
FileHandleWithStandardError = FileHandleRef
FileHandleWithStandardInput = FileHandleRef
FileHandleWithStandardOutput = FileHandleRef
FileHandleWithNullDevice = FileHandleRef
```

#### Create

```
FileHandleWithFileDescriptor( long fd ) = FileHandleRef
FileHandleWithFileDescriptorCloseOnDealloc( long fd, Boolean flag ) = FileHandleRef
```

#### File descriptor

```
FileHandleFileDescriptor( FileHandleRef fh ) = long
```

#### Reading

```
FileHandleAvailableData( FileHandleRef fh ) = CFDataRef
FileHandleReadDataToEndOfFile( FileHandleRef fh ) = CFDataRef
FileHandleReadDataOfLength( FileHandleRef fh, NSUInteger length ) = CFDataRef
```

#### Writing

```
FileHandleWriteData( FileHandleRef fh, CFDataRef dta )
```

#### Reading/writing using blocks

```
FileHandleSetReadabilityHandler( FileHandleRef fh, ptr handlerFn )
FileHandleSetWriteabilityHandler( FileHandleRef fh, ptr handlerFn )
```

#### Communicating asynchronously

```
FileHandleAcceptConnectionInBackgroundAndNotify( FileHandleRef fh )
FileHandleAcceptConnectionInBackgroundAndNotifyForModes( FileHandleRef fh, CFArrayRef modes )
FileHandleReadInBackgroundAndNotify( FileHandleRef fh )
FileHandleReadInBackgroundAndNotifyForModes( FileHandleRef fh, CFArrayRef modes )
FileHandleReadToEndOfFileInBackgroundAndNotify( FileHandleRef fh )
FileHandleReadToEndOfFileInBackgroundAndNotifyForModes( FileHandleRef fh, CFArrayRef modes )
FileHandleWaitForDataInBackgroundAndNotify( FileHandleRef fh )
FileHandleWaitForDataInBackgroundAndNotifyForModes( FileHandleRef fh, CFArrayRef modes )
```

#### Seeking within a file

```
FileHandleOffsetInFile( FileHandleRef fh ) = UInt64
FileHandleSeekToEndOfFile( FileHandleRef fh ) = UInt64
FileHandleSeekToFileOffset( FileHandleRef fh, UInt64 offset )
```

#### Operating on a file

```
FileHandleCloseFile( FileHandleRef fh )
FileHandleSynchronizeFile( FileHandleRef fh )
FileHandleTruncateFileAtOffset( FileHandleRef fh, UInt64 offset )
```

### Apple documentation

[NSFileHandle](#)



## FileManager

### Functions

Create

```
FileManagerDefaultManager = FileManagerRef
```

User directories

```
FileManagerTemporaryDirectory = CFURLRef
```

```
FileManagerHomeDirectoryForCurrentUser = CFURLRef
```

```
FileManagerHomeDirectoryForUser( CFStringRef userName ) = CFURLRef
```

System directories

```
FileManagerURLForDirectoryAppropriateForURL( NSSearchPathDirectory whichDirectory, NSSearchPathDomainMask domain, CFURLRef url, Boolean create, ErrorRef *err ) = CFURLRef
```

```
FileManagerURLForDirectory( NSSearchPathDirectory whichDirectory, NSSearchPathDomainMask domain ) = CFURLRef
```

```
FileManagerURLsForDirectory( NSSearchPathDirectory whichDirectory, NSSearchPathDomainMask domain ) = CFArrayRef
```

Directory contents

```
FileManagerContentsOfDirectoryAtURL( CFURLRef url, CFArrayRef keys, NSDirectoryEnumerationOptions options ) = CFArrayRef
```

```
FileManagerEnumeratorAtURL( CFURLRef url, CFArrayRef propertyKeys, NSDirectoryEnumerationOptions options, ptr errorCallback, ptr userData ) = DirectoryEnumeratorRef
```

```
FileManagerMountedVolumeURLs( CFArrayRef propertyKeys, NSVolumeEnumerationOptions options ) = CFArrayRef
```

Creating and deleting items

```
FileManagerCreateDirectoryAtURL( CFURLRef url, Boolean withIntermediateDirectories, CFDictionaryRef attributes ) = Boolean
```

```
FileManagerCreateFileAtURL( CFURLRef url, CFDataRef contents, CFDictionaryRef attributes ) = Boolean
```

```
FileManagerRemoveItemAtURL( CFURLRef url ) = Boolean
```

```
FileManagerReplaceItemAtURL( CFURLRef origURL, CFURLRef newURL ) = Boolean
```

```
FileManagerTrashItemAtURL( CFURLRef url )
```

Moving and copying items

```
FileManagerCopyItemAtURL( CFURLRef srcURL, CFURLRef dstURL ) = Boolean
```

```
FileManagerMoveItemAtURL( CFURLRef srcURL, CFURLRef dstURL ) = Boolean
```

iCloud-based items

```
FileManagerUbiquityIdentityToken = CTypeRef// macOS 10.8+
```

```
FileManagerURLForUbiquityContainerIdentifier( CFStringRef identifier ) = CFURLRef
```

```
FileManagerIsUbiquitousItem( CFURLRef url ) = Boolean
```

```
FileManagerSetUbiquitousItem( Boolean flag, CFURLRef url, CFURLRef destURL, ErrorRef *err ) = Boolean
```

```
FileManagerStartDownloadingUbiquitousItem( CFURLRef url, ErrorRef *err ) = Boolean
```

```
FileManagerEvictUbiquitousItem( CFURLRef url, ErrorRef *err ) = Boolean
```

```
FileManagerURLForPublishingUbiquitousItem( CFURLRef url, CFDateRef *expirationDate, ErrorRef *err ) = CFURLRef
```

File provider services

```
FileManagerGetFileProviderServicesForItem( CFURLRef url, ptr completionHandler, ptr userData )// macOS 10.13+
```

Symbolic and hard links

```
FileManagerCreateSymbolicLinkAtURL( CFURLRef url, CFURLRef dstURL, ErrorRef *err ) = Boolean
```

```
FileManagerLinkItemAtURL( CFURLRef srcURL, CFURLRef dstURL, ErrorRef *err ) = Boolean
```

```
FileManagerDestinationURLOfSymbolicLinkAtURL( CFURLRef url, ErrorRef *err ) = CFURLRef
```

Determining access

```
FileManagerFileExistsAtURL( CFURLRef url ) = Boolean
```

```
FileManagerIsReadableFileAtURL( CFURLRef url ) = Boolean
```

```
FileManagerIsWritableFileAtURL( CFURLRef url ) = Boolean
```

```
FileManagerIsExecutableFileAtURL( CFURLRef url ) = Boolean
```

```
FileManagerIsDeletableFileAtURL( CFURLRef url ) = Boolean
```

Attributes

```
FileManagerAttributesOfItemAtURL( CFURLRef url ) = CFDictionaryRef
```

```
FileManagerSetAttributesOfItemAtURL( CFURLRef url, CFDictionaryRef attributes ) = Boolean
```

File contents

```
FileManagerContentsAtURL( CFURLRef url ) = CFDataRef
```

```
FileManagerContentsEqual( CFURLRef url1, CFURLRef url2 ) = Boolean
```

Relationship between items

```
FileManagerGetRelationshipOfDirectoryAtURL( NSURLRelationship *relationship, CFURLRef dirURL, CFURLRef itemURL, ErrorRef *err ) = Boolean// macOS 10.10+
```

```
FileManagerGetRelationshipOfDirectoryInDomain( NSURLRelationship *relationship, NSSearchPathDirectory whichDirectory, NSSearchPathDomainMask domain, CFURLRef itemURL, ErrorRef *err ) = Boolean// macOS 10.10+
```

Current directory

```
FileManagerChangeCurrentDirectoryURL( CFURLRef url ) = Boolean  
FileManagerCurrentDirectoryURL = CFURLRef
```

Unmount volumes

```
FileManagerUnmountVolumeAtURL( CFURLRef url, NSFileManagerUnmountOptions options, ptr completionHandler, ptr  
userData )// mac OS 10.11+
```

Convenience

```
FileManagerIsDirectoryAtURL( CFURLRef url ) = Boolean  
FileManagerURLForPreferencesDirectory = CFURLRef  
FileManagerURLForApplicationDirectory = CFURLRef
```

## Apple documentation

[NSFileManager](#)



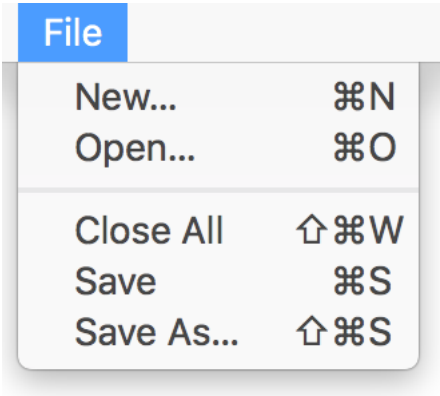


FileMenu

statement

**Syntax**  
`filemenu menuIndex`

**Description**  
Builds a default file menu. Requires the Cocoa runtime ([CocoaInit](#)).





---

## FileWrapper

### Functions

#### Create

```
FileWrapperWithURL( CFURLRef url, NSFileWrapperReadingOptions options, ErrorRef *err ) = FileWrapperRef
FileWrapperDirectoryWithFileWrappers( CFDictionaryRef childrenByPreferredName ) = FileWrapperRef
FileWrapperRegularFileWithContents( CFDataRef contents ) = FileWrapperRef
FileWrapperSymbolicLinkWithDestinationURL( CFURLRef url ) = FileWrapperRef
FileWrapperWithSerializedRepresentation( CFDataRef serializedRepresentation ) = FileWrapperRef
```

#### Query

```
FileWrapperIsRegularFile( FileWrapperRef ref ) = Boolean
FileWrapperIsDirectory( FileWrapperRef ref ) = Boolean
FileWrapperIsSymbolicLink( FileWrapperRef ref ) = Boolean
```

#### Info

```
FileWrapperFileWrappers( FileWrapperRef ref ) = CFDictionaryRef
FileWrapperAddFileWrapper( FileWrapperRef ref, FileWrapperRef child ) = CFStringRef
FileWrapperRemoveFileWrapper( FileWrapperRef ref, FileWrapperRef child )
FileWrapperAddRegularFileWithContents( FileWrapperRef ref, CFDataRef dta, CFStringRef preferredFilename ) = CFStringRef
FileWrapperKeyForFileWrapper( FileWrapperRef ref, FileWrapperRef child ) = CFStringRef
FileWrapperSymbolicLinkDestinationURL( FileWrapperRef ref ) = CFURLRef
```

#### Update

```
FileWrapperMatchesContentsOfURL( FileWrapperRef ref, CFURLRef url ) = Boolean
FileWrapperReadFromURL( FileWrapperRef ref, CFURLRef url, NSFileWrapperReadingOptions options, ErrorRef *err ) = Boolean
```

#### Serialize

```
FileWrapperSerializedRepresentation( FileWrapperRef ref ) = CFDataRef
```

#### Files

```
FileWrapperFilename( FileWrapperRef ref ) = CFStringRef
FileWrapperPreferredFilename( FileWrapperRef ref ) = CFStringRef
FileWrapperFileAttributes( FileWrapperRef ref ) = CFDictionaryRef
FileWrapperRegularFileContents( FileWrapperRef ref ) = CFDataRef
```

#### Write files

```
FileWrapperWriteToURL( FileWrapperRef ref, CFURLRef url, NSFileWrapperWritingOptions options, CFURLRef origContentsURL, ErrorRef *err ) = Boolean
```

#### Icons

```
FileWrapperIcon( FileWrapperRef ref ) = ImageRef
```

### Apple documentation

[NSFileWrapper](#)

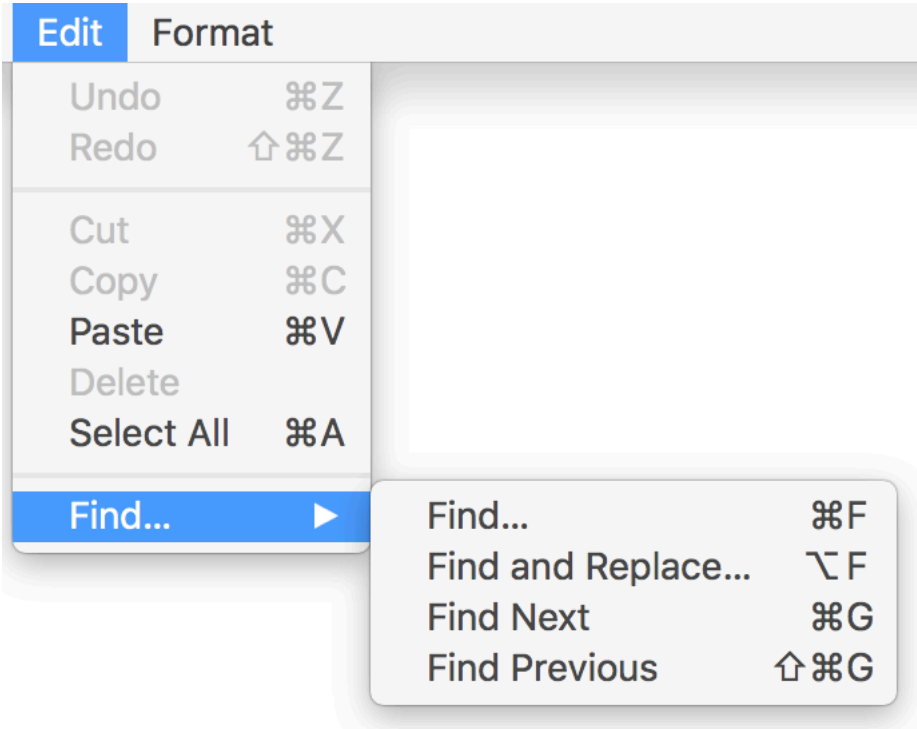


FindMenu

statement

**Syntax**  
`findmenu menuIndex`

**Description**  
Builds a default find menu. Requires the Cocoa runtime ([CocoaInit](#)).





## Font

### Functions

Arbitrary font

```
FontWithName( CFStringRef name, CGFloat size ) = FontRef
FontWithDescriptor( FontDescriptorRef descriptor, CGFloat size ) = FontRef
FontWithDescriptorAndTextTransform( FontDescriptorRef descriptor, AffineTransformRef tx ) = FontRef
FontWithNameAndMatrix( CFStringRef name, CGFloat *matrix ) = FontRef
```

User font

```
FontUserFontOfSize( CGFloat size ) = FontRef// if size is 0 or negative, returns the user font at the default size
FontUserFixedPitchFontOfSize( CGFloat size ) = FontRef// if size is 0 or negative, returns the fixed-pitch font at the default size
```

System font

```
FontBoldSystemFontOfSize( CGFloat size ) = FontRef
FontControlContentFontOfSize( CGFloat size ) = FontRef
FontLabelFontOfSize( CGFloat size ) = FontRef
FontMenuFontOfSize( CGFloat size ) = FontRef
FontMenuBarFontOfSize( CGFloat size ) = FontRef
FontMessageFontOfSize( CGFloat size ) = FontRef
FontMonospacedDigitSystemFontOfSize( CGFloat size, CGFloat weight ) = FontRef// macOS 10.11+
FontPaletteFontOfSize( CGFloat size ) = FontRef
FontSystemFontOfSize( CGFloat size ) = FontRef
FontSystemFontOfSizeAndWeight( CGFloat size, CGFloat weight ) = FontRef// macOS 10.11+
FontTitleBarFontOfSize( CGFloat size ) = FontRef
FontToolTipsFontOfSize( CGFloat size ) = FontRef
```

Draw

```
FontSet( FontRef font )
FontSetInContext( FontRef font, GraphicsContextRef context )
```

General information

```
FontDescriptor( FontRef font ) = FontDescriptorRef
```

Metrics information

```
FontLabelFontSize = CGFloat
FontSmallSystemFontSize = CGFloat
FontSystemFontSize = CGFloat
FontSystemFontSizeForControlSize( NSControlSize size ) = CGFloat
FontAscender( FontRef font ) = CGFloat
FontBoundingRectForFont( FontRef font ) = CGRect
FontCapHeight( FontRef font ) = CGFloat
FontDescender( FontRef font ) = CGFloat
FontItalicAngle( FontRef font ) = CGFloat
FontLeading( FontRef font ) = CGFloat
FontMatrix( FontRef font ) = ptr
FontSize( FontRef font ) = CGFloat
FontUnderlinePosition( FontRef font ) = CGFloat
FontUnderlineThickness( FontRef font ) = CGFloat
FontXHeight( FontRef font ) = CGFloat
```

Font names

```
FontDisplayName( FontRef font ) = CFStringRef
FontFamilyName( FontRef font ) = CFStringRef
FontName( FontRef font ) = CFStringRef
```

Set user font

```
FontSetUserFont( FontRef font )
```

Vertical fonts

```
FontIsVertical( FontRef font ) = Boolean
FontVerticalFont( FontRef font ) = FontRef
```

### Apple documentation

[NSFont](#)



---

## FontDescriptor

### Functions

#### Create

```
FontDescriptorWithAttributes( CFDictionaryRef attributes ) = FontDescriptorRef
FontDescriptorWithNameAndMatrix( CFStringRef name, AffineTransformRef matrix ) = FontDescriptorRef
FontDescriptorWithNameAndSize( CFStringRef name, CGFloat size ) = FontDescriptorRef
FontDescriptorByAddingAttributes( FontDescriptorRef fd, CFDictionaryRef attrs ) = FontDescriptorRef
FontDescriptorWithFace( FontDescriptorRef fd, CFStringRef face ) = FontDescriptorRef
FontDescriptorWithFamily( FontDescriptorRef fd, CFStringRef family ) = FontDescriptorRef
FontDescriptorWithMatrix( FontDescriptorRef fd, AffineTransformRef matrix ) = FontDescriptorRef
FontDescriptorWithSize( FontDescriptorRef fd, CGFloat size ) = FontDescriptorRef
FontDescriptorWithSymbolicTraits( FontDescriptorRef fd, UInt32 traits ) = FontDescriptorRef
```

#### Finding fonts

```
FontDescriptorMatchingDescriptorsWithMandatoryKeys( FontDescriptorRef fd, CFSetRef keys ) = CFArrayRef
FontDescriptorMatchingDescriptorWithMandatoryKeys( FontDescriptorRef fd, CFSetRef keys ) = FontDescriptorRef
```

#### Attributes

```
FontDescriptorAttributes( FontDescriptorRef fd ) = CFDictionaryRef
FontDescriptorObjectForKey( FontDescriptorRef fd, CFStringRef attribute ) = CFTypeRef
FontDescriptorMatrix( FontDescriptorRef fd ) = AffineTransformRef
FontDescriptorPointSize( FontDescriptorRef fd ) = CGFloat
FontDescriptorPostscriptName( FontDescriptorRef fd ) = CFStringRef
```

#### Traits

```
FontDescriptorSymbolicTraits( FontDescriptorRef fd ) = UInt32
```

#### Assets

```
FontDescriptorRequiresFontAssetRequest( FontDescriptorRef fd ) = Boolean
```

### Apple documentation

[NSFontDescriptor](#)



## FontManager

### Description

Functions for managing the font panel.

### Dialog Events

`_fontManagerChangeFont`

### Functions

Shared

`FontManagerShared = FontManagerRef`

Default conversion

`FontManagerSetFontManagerFactory( ClassRef class )`

`FontManagerSetFontPanelFactory( ClassRef class )`

Available fonts

`FontManagerAvailableFonts = CFArrayRef`

`FontManagerAvailableFontFamilies = CFArrayRef`

`FontManagerAvailableFontNamesWithTraits( NSFontTraitMask traits ) = CFArrayRef`

`FontManagerAvailableMembersOfFontFamily( CFStringRef fam ) = CFArrayRef`

Selected font

`FontManagerSetSelectedFont( CTNSFontRef font )`

`FontManagerIsMultiple = Boolean`

`FontManagerSelectedFont = FontRef`

`FontManagerSendAction = Boolean`

`FontManagerLocalizedStringForFamily( CFStringRef family, CFStringRef faceKey ) = CFStringRef`

Sending actions

`FontManagerAddFontTrait`

`FontManagerRemoveFontTrait`

`FontManagerModifyFont`

`FontManagerModifyFontViaPanel`

`FontManagerOrderFrontStylesPanel`

`FontManagerOrderFrontFontPanel`

Converting automatically

`FontManagerConvertFont( FontRef font ) = FontRef`

Convert manually

`FontManagerConvertFontToFace( FontRef font, CFStringRef face ) = FontRef`

`FontManagerConvertFontToFamily( FontRef font, CFStringRef family ) = FontRef`

`FontManagerConvertFontToHaveTrait( FontRef font, NSFontTraitMask traitMask ) = FontRef`

`FontManagerConvertFontToNotHaveTrait( FontRef font, NSFontTraitMask traitMask ) = FontRef`

`FontManagerConvertFontToSize( FontRef font, CGFloat size ) = FontRef`

`FontManagerConvertFontWeight( FontRef font, Boolean upFlag ) = FontRef`

`FontManagerCurrentFontAction = NSFontAction`

`FontManagerConvertFontTraits( NSFontTraitMask traitMask ) = NSFontTraitMask`

Get font

`FontManagerFontWithFamily( CFStringRef family, NSFontTraitMask traitMask, NSInteger weight, CGFloat size ) = FontRef`

Examining fonts

`FontManagerTraitsOfFont( FontRef font ) = NSFontTraitMask`

`FontManagerFontNamedHasTraits( CFStringRef name, NSFontTraitMask traitMask ) = Boolean`

`FontManagerWeightOfFont( FontRef font ) = NSInteger`

Managing the font panel and font menu

`FontManagerIsEnabled = Boolean`

`FontManagerFontPanel( Boolean create ) = FontPanelRef`

`FontManagerFontMenu( Boolean create ) = CocoaMenuRef`

`FontManagerSetFontMenu( CocoaMenuRef m )`

Action and target

`FontManagerAction = SELRef`

`FontManagerTarget = CFTyperef`

Attributes

`FontManagerSetSelectedAttributes( CFDictionaryRef attrs, Boolean isMultiple )`

`FontManagerConvertAttributes( CFDictionaryRef attrs ) = CFDictionaryRef`

Convenience

[FontManagerShowPanel](#)

[FontManagerClosePanel](#)

## **Apple documentation**

[NSFontManager](#)

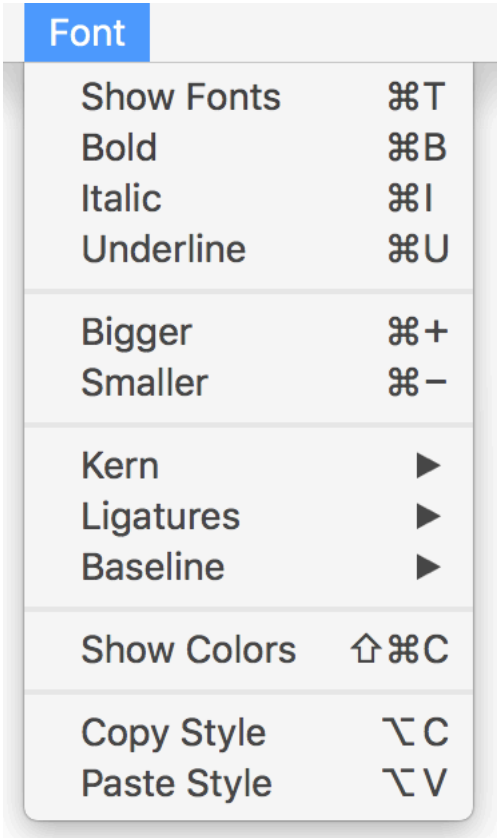


FontMenu

statement

**Syntax**  
`fontmenu menuIndex`

**Description**  
Builds a default font menu. Requires the Cocoa runtime ([CocoaInit](#)).







Format ▶

## FormatMenu

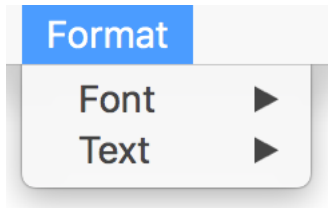
statement

### Syntax

**formatmenu** *menuIndex*

### Description

Builds a default format menu. Requires the Cocoa runtime ([CocoaInit](#)).





---

### Geometry

#### Functions

```
StringFromPoint( CGPoint pt ) = CFStringRef
StringFromSize( CGSize size ) = CFStringRef
StringFromRect( CGRect rect ) = CFStringRef
PointFromString( CFStringRef string ) = CGPoint
SizeFromString( CFStringRef string ) = CGSize
RectFromString( CFStringRef string ) = CGRect
```



---

## GestureRecognizer

### Functions

#### Init

```
GestureRecognizerInit( ptr callback, ptr userData ) = GestureRecognizerRef// autoreleased
```

#### Location of events

```
GestureRecognizerLocationInView( GestureRecognizerRef ref, NSInteger tag ) = CGPoint
```

#### State

```
GestureRecognizerState( GestureRecognizerRef ref ) = NSGestureRecognizerState
```

```
GestureRecognizerView( GestureRecognizerRef ref ) = NSInteger
```

```
GestureRecognizerIsEnabled( GestureRecognizerRef ref ) = Boolean
```

```
GestureRecognizerSetEnabled( GestureRecognizerRef ref, Boolean flag )
```

#### Delaying

```
GestureRecognizerDelaysPrimaryMouseButtonEvents( GestureRecognizerRef ref ) = Boolean
```

```
GestureRecognizerSetDelaysPrimaryMouseButtonEvents( GestureRecognizerRef ref, Boolean flag )
```

```
GestureRecognizerDelaysSecondaryMouseButtonEvents( GestureRecognizerRef ref ) = Boolean
```

```
GestureRecognizerSetDelaysSecondaryMouseButtonEvents( GestureRecognizerRef ref, Boolean flag )
```

```
GestureRecognizerDelaysOtherMouseButtonEvents( GestureRecognizerRef ref ) = Boolean
```

```
GestureRecognizerSetDelaysOtherMouseButtonEvents( GestureRecognizerRef ref, Boolean flag )
```

```
GestureRecognizerDelaysKeyEvents( GestureRecognizerRef ref ) = Boolean
```

```
GestureRecognizerSetDelaysKeyEvents( GestureRecognizerRef ref, Boolean flag )
```

```
GestureRecognizerDelaysMagnificationEvents( GestureRecognizerRef ref ) = Boolean
```

```
GestureRecognizerSetDelaysMagnificationEvents( GestureRecognizerRef ref, Boolean flag )
```

```
GestureRecognizerDelaysRotationEvents( GestureRecognizerRef ref ) = Boolean
```

```
GestureRecognizerSetDelaysRotationEvents( GestureRecognizerRef ref, Boolean flag )
```

#### Pressure

```
GestureRecognizerPressureConfiguration( GestureRecognizerRef ref ) = PressureConfigurationRef// macOS 10.11+
```

```
GestureRecognizerSetPressureConfiguration( GestureRecognizerRef ref, PressureConfigurationRef config )// macOS 10.11+
```

#### Instance properties

```
GestureRecognizerAllowedTouchTypes( GestureRecognizerRef ref ) = NSTouchTypeMask// macOS 10.12.2+
```

```
GestureRecognizerSetAllowedTouchTypes( GestureRecognizerRef ref, NSTouchTypeMask touchTypes )// macOS 10.12.2+
```

### Apple documentation

[NSGestureRecognizer](#)



---

## GlyphGenerator

### Functions

Obtaining

```
GlyphGeneratorShared = GlyphGeneratorRef
```

Generating glyphs

```
GlyphGeneratorGenerateGlyphsForGlyphStorage( GlyphGeneratorRef ref, GlyphStorageRef storage, NSUInteger  
desiredNumberOfCharacters, NSUInteger *glyphIndex, NSUInteger *characterIndex )
```

### Apple documentation

[NSGlyphGenerator](#)



---

## GlyphStorage

### Functions

TextStorage

```
GlyphStorageAttributedString( GlyphStorageRef ref ) = CFAttributedStringRef
```

Display options

```
GlyphStorageLayoutOptions( GlyphStorageRef ref ) = UInt32
```

Cache

```
GlyphStorageInsertGlyphs( GlyphStorageRef ref, NSRange *glyphs, NSUInteger length, NSUInteger glyphIndex, NSUInteger charIndex )
```

```
GlyphStorageSetIntAttribute( GlyphStorageRef ref, NSInteger attributeTag, NSInteger value, NSUInteger glyphIndex )
```

### Apple documentation

[NSGlyphStorage](#)



---

## Gradient

### Functions

Init

```
GradientWithStartEndColors( ColorRef startColor, ColorRef endColor ) = GradientRef
GradientWithColors( CFArrayRef colors ) = GradientRef
GradientWithColorsAndLocations( ColorRef firstColor, ... ) = GradientRef
GradientWithColorsAtLocations( CFArrayRef colors, CGFloat *locations, ColorSpaceRef cs ) = GradientRef
```

Primitive drawing functions

```
GradientDrawFromPoint( GradientRef ref, CGPoint pt1, CGPoint pt2, NSGradientDrawingOptions options )
GradientDrawFromCenter( GradientRef ref, CGPoint startCenter, CGFloat startRadius, CGPoint endCenter, CGFloat
endRadius, NSGradientDrawingOptions options )
```

Drawing linear gradients

```
GradientDrawInRect( GradientRef ref, CGRect rect, CGFloat angle )
GradientDrawInBezierPath( GradientRef ref, BezierPathRef path, CGFloat angle )
```

Drawing radial gradients

```
GradientDrawInRectRelativeCenterPosition( GradientRef ref, CGRect rect, CGPoint position )
GradientDrawInBezierPathRelativeCenterPosition( GradientRef ref, BezierPathRef path, CGPoint position )
```

Gradient properties

```
GradientColorSpace( GradientRef ref ) = ColorSpaceRef
GradientNumberOfColorStops( GradientRef ref ) = NSInteger
GradientGetColor( GradientRef ref, ColorRef *col, CGFloat *location, NSInteger index )
GradientInterpolatedColorAtLocation( GradientRef ref, CGFloat location ) = ColorRef
```

Convenience

```
GradientDrawFromPointWithStartEndColors( CGPoint pt1, CGPoint pt2, ColorRef startColor, ColorRef endColor,
NSGradientDrawingOptions options )
GradientDrawFromCenterWithStartEndColors( CGPoint startCenter, CGFloat startRadius, CGPoint endCenter, CGFloat
endRadius, ColorRef startColor, ColorRef endColor, NSGradientDrawingOptions options )
GradientDrawInRectWithStartEndColors( CGRect rect, CGFloat angle, ColorRef startColor, ColorRef endColor )
GradientDrawInBezierPathWithStartEndColors( BezierPathRef path, CGFloat angle, ColorRef startColor, ColorRef
endColor )
GradientDrawInRectRelativeCenterPositionWithStartEndColors( CGRect rect, CGPoint position, ColorRef startColor,
ColorRef endColor )
GradientDrawInBezierPathRelativeCenterPositionWithStartEndColors( BezierPathRef path, CGPoint position, ColorRef
startColor, ColorRef endColor )
```

### Apple documentation

[NSGradient](#)



---

## Graphics

### Functions

#### Rectangles

```
NSRectFill( CGRect r )
NSRectFillList( CGRect *rects, NSInteger count )
NSRectFillListWithGrays( CGRect *rects, CGFloat *grays, NSInteger count )
NSRectFillListWithColors( CGRect *rects, ColorRef *cols, NSInteger count )
NSRectFillListUsingOperation( CGRect *rects, NSInteger count, NSCompositingOperation op )
NSRectFillListWithColorsUsingOperation( CGRect *rects, ColorRef *cols, NSInteger count, NSCompositingOperation op )
NSRectFillUsingOperation( CGRect r, NSCompositingOperation operation )
NSFrameRect( CGRect r )
NSFrameRectWithWidth( CGRect r, CGFloat frameWidth )
NSFrameRectWithWidthUsingOperation( CGRect r, CGFloat frameWidth, NSCompositingOperation operation )
NSEraseRect( CGRect r )
NSSetFocusRingStyle( NSFocusRingPlacement placement )
```

#### Window depth

```
NSBestDepth( CFStringRef colorSpaceName, NSInteger bps, NSInteger bpp, Boolean planar, Boolean *exactMatch ) =
NSWindowDepth
NSPlanarFromDepth( NSWindowDepth depth ) = Boolean
NSColorSpaceFromDepth( NSWindowDepth depth ) = CFStringRef
NSBitsPerSampleFromDepth( NSWindowDepth depth ) = NSInteger
NSBitsPerPixelFromDepth( NSWindowDepth depth ) = NSInteger
NSNumberOfColorComponents( CFStringRef colorSpaceName ) = NSInteger
```

#### System animations

```
NSShowAnimationEffect( NSAnimationEffect animationEffect, CGPoint centerLocation, CGSize size, ptr didEndCallback,
ptr contextInfo )
```

#### Other functions

```
NSDrawThreePartImage( CGRect frame, ImageRef startCap, ImageRef centerFill, ImageRef endCap, Boolean vertical,
NSCompositingOperation op, CGFloat alphaFraction, Boolean flipped )
NSDrawNinePartImage( CGRect frame, ImageRef tl, ImageRef t, ImageRef tr, ImageRef l, ImageRef c, ImageRef r,
ImageRef bl, ImageRef b, ImageRef br, NSCompositingOperation op, CGFloat alphaFraction, Boolean flipped )
```



---

## GraphicsContext

### Description

General functions to work with graphics.

### Functions

Create

```
GraphicsContextWithAttributes( CFDictionaryRef attributes ) = GraphicsContextRef
GraphicsContextWithBitmapImageRep( BitmapImageRepRef rep ) = GraphicsContextRef
GraphicsContextWithCGContext( CGContextRef cgCtx, Boolean flipped ) = GraphicsContextRef // macOS 10.10+
GraphicsContextWithWindow( CocoaWindowRef window ) = GraphicsContextRef
```

Current context

```
GraphicsContextCurrentContext = GraphicsContextRef
GraphicsContextSetCurrentContext( GraphicsContextRef ctx )
GraphicsContextCurrentCGContext = CGContextRef
```

Graphics state

```
GraphicsContextSaveGraphicsState
GraphicsContextRestoreGraphicsState
```

Testing drawing destination

```
GraphicsContextCurrentContextDrawingToScreen = Boolean
GraphicsContextDrawingToScreen( GraphicsContextRef ctx ) = Boolean
```

Context info

```
GraphicsContextAttributes( GraphicsContextRef ctx ) = CFDictionaryRef
GraphicsContextIsFlipped( GraphicsContextRef ctx ) = Boolean
```

Flushing

```
GraphicsContextFlushGraphics( GraphicsContextRef ctx )
```

Rendering options

```
GraphicsContextCompositingOperation( GraphicsContextRef ctx ) = NSCompositingOperation
GraphicsContextSetCompositingOperation( GraphicsContextRef ctx, NSCompositingOperation operation )
GraphicsContextImageInterpolation( GraphicsContextRef ctx ) = NSImageInterpolation
GraphicsContextSetImageInterpolation( GraphicsContextRef ctx, NSImageInterpolation interpolation )
GraphicsContextShouldAntialias( GraphicsContextRef ctx ) = Boolean
GraphicsContextSetShouldAntialias( GraphicsContextRef ctx, Boolean flag )
GraphicsContextPatternPhase( GraphicsContextRef ctx ) = CGPoint
GraphicsContextSetPatternPhase( GraphicsContextRef ctx, CGPoint phase )
```

CIContext

```
GraphicsContextCIContext( GraphicsContextRef ctx ) = CIContextRef
```

Color rendering intent

```
GraphicsContextColorRenderingIntent( GraphicsContextRef ctx ) = NSColorRenderingIntent
GraphicsContextSetColorRenderingIntent( GraphicsContextRef ctx, NSColorRenderingIntent intent )
```

### Apple documentation

[NSGraphicsContext](#)





---

## HelpManager

### Functions

Get

```
HelpManagerShared = HelpManagerRef
```

Display

```
HelpManagerFindString( CFStringRef string, CFStringRef book )
```

```
HelpManagerOpenHelpAnchor( CFStringRef anchor, CFStringRef book )
```

Add help books

```
HelpManagerRegisterBooksInBundle( BundleRef bundle ) = Boolean
```

Configure

```
HelpManagerSetContextHelp( CFAttributedStringRef attrString, ObjectRef object )
```

```
HelpManagerRemoveContextHelp( ObjectRef object )
```

Display

```
HelpManagerContextHelp( ObjectRef object ) = CFAttributedStringRef
```

```
HelpManagerShowContextHelp( ObjectRef object, CGPoint locationHint ) = Boolean
```

Properties

```
HelpManagerIsContextHelpModeActive = Boolean// macOS 10.13+
```

### Apple documentation

[NSHelpManager](#)



## HelpMenu

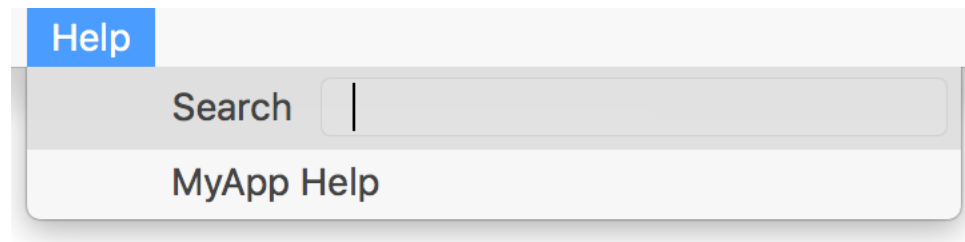
statement

### Syntax

**helpmenu** *menuIndex*

### Description

Builds a default help menu. Requires the Cocoa runtime ([CocoaInit](#)).





## Image

### Functions

Image by name

```
ImageNamed( CFStringRef name ) = ImageRef
ImageSetName( ImageRef ref, CFStringRef name )
ImageName( ImageRef ref ) = CFStringRef
```

Dynamically drawn image

```
ImageWithDrawingHandler( CGSize size, Boolean flipped, ptr handler, ptr userData ) = ImageRef
```

Image from resource file

```
ImageByReferencingURL( CFURLRef url ) = ImageRef
ImageWithSize( CGSize size ) = ImageRef
```

Image from existing data

```
ImageWithData( CFDataRef dta ) = ImageRef
ImageWithDataIgnoringOrientation( CFDataRef dta ) = ImageRef
ImageWithCGImage( CGImageRef image, CGSize size ) = ImageRef
ImageWithPasteboard( CocoaPasteboardRef pb ) = ImageRef
```

Create empty image

```
ImageWithSize( CGSize size ) = ImageRef
```

Attributes

```
ImageSize( ImageRef ref ) = CGSize
ImageSetSize( ImageRef ref, CGSize size )
ImageSetTemplate( ImageRef ref, Boolean flag )
ImageIsTemplate( ImageRef ref ) = Boolean
```

Image types

```
ImageTypes = CFArrayRef
ImageUnfilteredTypes = CFArrayRef
ImageCanInitWithPasteboard( CocoaPasteboardRef pb ) = Boolean
```

Image representation

```
ImageAddRepresentation( ImageRef ref, ImageRepRef rep )
ImageAddRepresentations( ImageRef ref, CFArrayRef reps )
ImageRepresentations( ImageRef ref ) = CFArrayRef
ImageRemoveRepresentation( ImageRef ref, ImageRepRef rep )
ImageBestRepresentationForRect( ImageRef ref, CGRect r, GraphicsContextRef ctx, CFDictionaryRef hints ) = ImageRepRef
```

Image representation selection criteria

```
ImagePrefersColorMatch( ImageRef ref ) = Boolean
ImageSetPrefersColorMatch( ImageRef ref, Boolean flag )
ImageUsesEPSOnResolutionMismatch( ImageRef ref ) = Boolean
ImageSetUsesEPSOnResolutionMismatch( ImageRef ref, Boolean flag )
ImageMatchesOnMultipleResolution( ImageRef ref ) = Boolean
ImageSetMatchesOnMultipleResolution( ImageRef ref, Boolean flag )
```

Drawing

```
ImageDrawInRect( ImageRef ref, CGRect rect )
ImageDrawAtPoint( ImageRef ref, CGPoint pt, CGRect fromRect, NSCompositingOperation operation, CGFloat fraction )
ImageDrawInRectFromRect( ImageRef ref, CGRect rect, CGRect fromRect, NSCompositingOperation op, CGFloat fraction )
ImageDrawInRectFromRectRespectFlipped( ImageRef ref, CGRect rect, CGRect fromRect, NSCompositingOperation op, CGFloat fraction, Boolean respectFlipped, CFDictionaryRef hints )
ImageDrawRepresentationInRect( ImageRef ref, ImageRepRef rep, CGRect rect ) = Boolean
```

Drawing options

```
ImageIsValid( ImageRef ref ) = Boolean
ImageBackgroundColor( ImageRef ref ) = ColorRef
ImageSetBackgroundColor( ImageRef ref, ColorRef col )
ImageCapInsets( ImageRef ref ) = NSEdgeInsets// macOS 10.10+
ImageSetCapInsets( ImageRef ref, NSEdgeInsets insets )// macOS 10.10+
ImageResizingMode( ImageRef ref ) = NSImageResizingMode// macOS 10.10+
ImageSetResizingMode( ImageRef ref, NSImageResizingMode mode )// macOS 10.10+
```

Focus

```
ImageLockFocus( ImageRef ref )
ImageLockFocusFlipped( ImageRef ref, Boolean flag )
ImageUnlockFocus( ImageRef ref )
```

#### Alignment

```
ImageAlignmentRect( ImageRef ref ) = CGRect  
ImageSetAlignmentRect( ImageRef ref, CGRect rect )
```

#### Caching options

```
ImageCacheMode( ImageRef ref ) = NSImageCacheMode  
ImageSetCacheMode( ImageRef ref, NSImageCacheMode mode )  
ImageRecache( ImageRef ref )
```

#### TIFF

```
ImageTIFFRepresentation( ImageRef ref ) = CFDataRef  
ImageTIFFRepresentationUsingCompression( ImageRef ref, NSTIFFCompression compression, float factor ) = CFDataRef
```

#### CGImage

```
ImageCGImageForProposedRect( ImageRef ref, CGRect *rect, GraphicsContextRef context, CFDictionaryRef hints ) =  
CGImageRef
```

#### Incremental loads

```
ImageCancelIncrementalLoad( ImageRef ref )
```

#### Hit testing

```
ImageHitTestRect( ImageRef ref, CGRect testRectDestSpace, CGRect imageRectDestSpace, GraphicsContextRef ctx,  
CFDictionaryRef hints, Boolean flipped ) = Boolean
```

#### Image accessibility

```
ImageAccessibilityDescription( ImageRef ref ) = CFStringRef  
ImageSetAccessibilityDescription( ImageRef ref, CFStringRef description )
```

#### Core animation

```
ImageLayerContentsForContentsScale( ImageRef ref, CGFloat scale ) = CFTypeRef  
ImageRecommendedLayerContentsScale( ImageRef ref, CGFloat preferredScale ) = CGFloat
```

#### Axis matching

```
ImageMatchesOnlyOnBestFittingAxis( ImageRef ref ) = Boolean  
ImageSetMatchesOnlyOnBestFittingAxis( ImageRef ref, Boolean flag )
```

## Apple documentation

[NSImage](#)



---

## ImageRep

### Functions

Create

```
ImageRepImageRepsWithContentsOfURL( CFURLRef url ) = CFArrayRef  
ImageRepWithContentsOfURL( CFURLRef url ) = ImageRepRef
```

Image types

```
ImageRepCanInitWithData( CFDataRef dta ) = Boolean  
ImageRepImageTypes = CFArrayRef  
ImageRepImageUnfilteredTypes = CFArrayRef
```

Image size

```
ImageRepSize( ImageRepRef rep ) = CGSize
```

Representation info

```
ImageRepBitsPerSample( ImageRepRef rep ) = NSInteger  
ImageRepColorSpaceName( ImageRepRef rep ) = CFStringRef  
ImageRepHasAlpha( ImageRepRef rep ) = Boolean  
ImageRepIsOpaque( ImageRepRef rep ) = Boolean  
ImageRepPixelsHigh( ImageRepRef rep ) = NSInteger  
ImageRepPixelsWide( ImageRepRef rep ) = NSInteger
```

CGImage

```
ImageRepCGImageForProposedRect( ImageRepRef rep, CGRect *r, GraphicsContextRef context, CFDictionaryRef hints ) =  
CGImageRef
```

Drawing

```
ImageRepDrawAtPoint( ImageRepRef rep, CGPoint pt ) = Boolean  
ImageRepDrawInRect( ImageRepRef rep, CGRect r ) = Boolean
```

### Apple documentation

[NSImageRep](#)



## ImageView

### Syntax

**imageview** *tag, enabled, image, rect, scaling, alignment, frameStyle, wndTag*

### Description

The **imageview** statement puts a new imageview in the current output cocoa window, or alters an existing imageview's characteristics.

### Parameters

<i>tag</i>	A number to identify the imageview. A negative tag hides the imageview
<i>enabled</i>	Enables or disables the imageview
<i>image</i>	This param can be NULL or one of the following: 1. The name of (or path to) an image resource. 2. An ImageRef (NSImage).
<i>rect</i>	Origin and size of the imageview in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>scaling</i>	The scaling mode applied to make the cell's image fit the frame of the imageview. <a href="#">NSImageScaleProportionallyDown</a> (default) <a href="#">NSImageScaleAxesIndependently</a> <a href="#">NSImageScaleNone</a> <a href="#">NSImageScaleProportionallyUpOrDown</a>
<i>alignment</i>	The alignment of the cell's image inside the imageview. <a href="#">NSImageAlignCenter</a> (default) <a href="#">NSImageAlignTop</a> <a href="#">NSImageAlignTopLeft</a> <a href="#">NSImageAlignTopRight</a> <a href="#">NSImageAlignLeft</a> <a href="#">NSImageAlignBottom</a> <a href="#">NSImageAlignBottomLeft</a> <a href="#">NSImageAlignBottomRight</a> <a href="#">NSImageAlignRight</a>
<i>frameStyle</i>	The style of the frame that appears around the image. <a href="#">NSImageFrameNone</a> (default) <a href="#">NSImageFramePhoto</a> <a href="#">NSImageFrameGrayBezel</a> <a href="#">NSImageFrameGroove</a> <a href="#">NSImageFrameButton</a>
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

Event Type	Description	ID
<i>_btnClick</i>	User dropped an image on the view.	Button tag

### Functions

```
ImageViewWithTag( NSInteger tag ) = ImageViewRef
ImageViewExists( NSInteger tag ) = Boolean
```

```
Init
ImageViewInit( NSInteger tag, CGRect r ) = ImageViewRef// autoreleased
```

```
Image
ImageViewImage( NSInteger tag ) = ImageRef
ImageViewSetImage( NSInteger tag, ImageRef image )
ImageViewSetImageNamed( NSInteger tag, CFStringRef imageName )
```

#### Characteristics

```
imageViewFrameStyle( NSInteger tag ) = NSImageFrameStyle
imageViewSetFrameStyle( NSInteger tag, NSImageFrameStyle style )
imageViewAlignment( NSInteger tag ) = NSImageAlignment
imageViewSetAlignment( NSInteger tag, NSImageAlignment alignment )
imageViewScaling( NSInteger tag ) = NSImageScaling
imageViewSetScaling( NSInteger tag, NSImageScaling scaling )
imageViewAnimates( NSInteger tag ) = Boolean
imageViewSetAnimates( NSInteger tag, Boolean flag )
```

#### Responding to user events

```
imageViewSetEditable( NSInteger tag, Boolean flag )
imageViewSetAllowsCutCopyPaste( NSInteger tag, Boolean flag )
```

#### See Also

[View](#)

#### Apple documentation

[NSImageView](#)



---

## IndexPath

### Functions

Create

```
IndexPathWithIndex( NSUInteger value ) = IndexPathRef  
IndexPathWithIndexes( NSUInteger *indexes, NSUInteger length ) = IndexPathRef
```

Special node names

```
IndexPathForItemInSection( NSInteger item, NSInteger section ) = IndexPathRef  
IndexPathSection( IndexPathRef path ) = NSInteger  
IndexPathRow( IndexPathRef path ) = NSInteger  
IndexPathItem( IndexPathRef path ) = NSInteger
```

Counting nodes

```
IndexPathLength( IndexPathRef path ) = NSInteger
```

Adding and removing nodes

```
IndexPathByAddingIndex( IndexPathRef path, NSUInteger index ) = IndexPathRef  
IndexPathByRemovingLastIndex( IndexPathRef path ) = IndexPathRef
```

Compare

```
IndexPathCompare( IndexPathRef path1, IndexPathRef path2 ) = NSComparisonResult
```

Working with indexes

```
IndexPathIndexAtPosition( IndexPathRef path, NSUInteger position ) = NSUInteger  
IndexPathGetIndexes( IndexPathRef path, NSUInteger *indexes, CFRange range )
```

### Apple documentation

[NSIndexPath](#)





## IndexSet

### Functions

#### Create

```
IndexSetWithIndex( NSUInteger value ) = IndexSetRef
IndexSetWithIndexesInRange( CFRange range ) = IndexSetRef
```

#### Query

```
IndexSetContainsIndex( IndexSetRef set, NSUInteger index ) = Boolean
IndexSetContainsIndexes( IndexSetRef set, IndexSetRef subset ) = Boolean
IndexSetContainsIndexesInRange( IndexSetRef set, CFRange range ) = Boolean
IndexSetIntersectsIndexesInRange( IndexSetRef set, CFRange range ) = Boolean
IndexSetCount( IndexSetRef set ) = NSUInteger
IndexSetCountOfIndexesInRange( IndexSetRef set, CFRange range ) = NSUInteger
IndexSetIndexPassingTest( IndexSetRef set, ptr predicate ) = NSUInteger
IndexSetIndexesPassingTest( IndexSetRef set, ptr predicate ) = IndexSetRef
IndexSetIndexWithOptionsPassingTest( IndexSetRef set, NSEnumerationOptions options, ptr predicate, ptr userData ) =
NSUInteger
IndexSetIndexesWithOptionsPassingTest( IndexSetRef set, NSEnumerationOptions options, ptr predicate, ptr userData )
= IndexSetRef
IndexSetIndexInRangePassingTest( IndexSetRef set, CFRange range, NSEnumerationOptions options, ptr predicate, ptr
userData ) = NSUInteger
IndexSetIndexesInRangePassingTest( IndexSetRef set, CFRange range, NSEnumerationOptions options, ptr predicate, ptr
userData ) = IndexSetRef
```

#### Enumerate content

```
IndexSetEnumerateRangesInRange( IndexSetRef set, CFRange range, NSEnumerationOptions options, ptr callback )
IndexSetEnumerateRanges( IndexSetRef set, ptr callback )
IndexSetEnumerateRangesWithOptions( IndexSetRef set, NSEnumerationOptions options, ptr callback )
```

#### Compare

```
IndexSetIsEqual( IndexSetRef set1, IndexSetRef set2 ) = Boolean
```

#### Getting indexes

```
IndexSetFirstIndex( IndexSetRef set ) = NSUInteger
IndexSetLastIndex( IndexSetRef set ) = NSUInteger
IndexSetIndexLessThan( IndexSetRef set, NSUInteger value ) = NSUInteger
IndexSetIndexLessThanOrEqualTo( IndexSetRef set, NSUInteger value ) = NSUInteger
IndexSetIndexGreaterThanOrEqualTo( IndexSetRef set, NSUInteger value ) = NSUInteger
IndexSetIndexGreaterThan( IndexSetRef set, NSUInteger value ) = NSUInteger
```

#### Enumerate indexes

```
IndexSetEnumerateIndexes( IndexSetRef set, ptr callback )
IndexSetEnumerateIndexesWithOptions( IndexSetRef set, NSEnumerationOptions options, ptr callback )
IndexSetEnumerateIndexesInRange( IndexSetRef set, CFRange range, NSEnumerationOptions options, ptr callback )
```

#### • Mutable index set •

```
IndexSetInit = MutableIndexSetRef// autoreleased
IndexSetAddIndex( MutableIndexSetRef set, NSUInteger index )
IndexSetAddIndexes( MutableIndexSetRef set, IndexSetRef otherSet )
IndexSetAddIndexesInRange( MutableIndexSetRef set, CFRange range )
IndexSetRemoveIndex( MutableIndexSetRef set, NSUInteger index )
IndexSetRemoveIndexes( MutableIndexSetRef set, IndexSetRef otherSet )
IndexSetRemoveAllIndexes( MutableIndexSetRef set )
IndexSetRemoveIndexesInRange( MutableIndexSetRef set, CFRange range )
IndexSetShiftIndexes( MutableIndexSetRef set, NSUInteger index, NSInteger delta )
```

### Apple documentation

[NSIndexSet](#)



---

## JSONSerialization

### Functions

Create object

```
JSONSerializationJSONObjectWithData( CFDataRef dta, NSJSONReadingOptions options, ErrorRef *err ) = CFTypeRef  
JSONSerializationJSONObjectWithStream( CFReadStreamRef stream, NSJSONReadingOptions options, ErrorRef *err ) =  
CFTypeRef
```

Create data

```
JSONSerializationDataWithJSONObject( CFTypeRef obj, NSJSONWritingOptions options, ErrorRef *err ) = CFDataRef  
JSONSerializationWriteJSONObjectToStream( CFTypeRef obj, CFWriteStreamRef stream, NSJSONWritingOptions options,  
ErrorRef *err ) = CFDataRef  
JSONSerializationIsValidJSONObject( CFTypeRef obj ) = Boolean
```

### Apple documentation

[NSJSONSerialization](#)



---

## KeyedArchiver

### Functions

Archiving data

```
KeyedArchiverArchivedDataWithRootObject( CTypeRef obj ) = CFDataRef
KeyedArchiverArchiveRootObjectToURL( CTypeRef obj, CFURLRef url ) = Boolean
KeyedArchiverFinishEncoding( KeyedArchiverRef ref )
KeyedArchiverOutputFormat( KeyedArchiverRef ref ) = NSPropertyListFormat
KeyedArchiverSetOutputFormat( KeyedArchiverRef ref, NSPropertyListFormat format )
KeyedArchiverRequiresSecureCoding( KeyedArchiverRef ref ) = Boolean// macOS 10.8+
KeyedArchiverSetRequiresSecureCoding( KeyedArchiverRef ref, Boolean flag )// macOS 10.9+ - Apple docs say macOS 10.8+ but clang throws up warning
```

Encoding

```
KeyedArchiverEncodeBool( KeyedArchiverRef ref, Boolean value, CFStringRef key )
KeyedArchiverEncodeBytes( KeyedArchiverRef ref, ptr bytes, NSUInteger length, CFStringRef key )
KeyedArchiverEncodeConditionalObject( KeyedArchiverRef ref, CTypeRef obj, CFStringRef key )
KeyedArchiverEncodeDouble( KeyedArchiverRef ref, double value, CFStringRef key )
KeyedArchiverEncodeFloat( KeyedArchiverRef ref, float value, CFStringRef key )
KeyedArchiverEncodeInt32( KeyedArchiverRef ref, SInt32 value, CFStringRef key )
KeyedArchiverEncodeInt64( KeyedArchiverRef ref, SInt64 value, CFStringRef key )
KeyedArchiverEncodeObject( KeyedArchiverRef ref, CTypeRef obj, CFStringRef key )
```

Classes and class names

```
KeyedArchiverSetClassNameForClass( CFStringRef codedName, ClassRef forClass )// class method
KeyedArchiverClassNameForClass( ClassRef forClass ) = CFStringRef// class method
KeyedArchiverSetClassName( KeyedArchiverRef ref, CFStringRef codedName, ClassRef forClass )// instance method
KeyedArchiverClassName( KeyedArchiverRef ref, ClassRef forClass ) = CFStringRef// instance method
```

Initializers

```
KeyedArchiverRequiringSecureCoding( Boolean flag ) = KeyedArchiverRef// macOS 10.14+ - Apple docs say macOS 10.13+ but clang throws up warning
```

Instance properties

```
KeyedArchiverEncodedData( KeyedArchiverRef ref ) = CFDataRef// macOS 10.12+
```

Type methods

```
KeyedArchiverArchivedDataRequiringSecureCoding( CTypeRef obj, Boolean secureCoding, ErrorRef *err ) = CFDataRef// macOS 10.13+
```

### See Also

[Keyedunarchiver](#)

### Apple documentation

[NSKeyedArchiver](#)



## KeyedUnarchiver

### Functions

Unarchiving data

```
KeyedUnarchiverUnarchiveObjectWithData( CFDataRef dta ) = CTypeRef
KeyedUnarchiverUnarchiveObjectWithURL( CFURLRef url ) = CTypeRef
KeyedUnarchiverRequiresSecureCoding( KeyedUnarchiverRef ref ) = Boolean// macOS 10.8+
KeyedUnarchiverSetRequiresSecureCoding( KeyedUnarchiverRef ref, Boolean flag )// macOS 10.8+
```

Decoding

```
KeyedUnarchiverContainsValue( KeyedUnarchiverRef ref, CFStringRef key ) = Boolean
KeyedUnarchiverDecodeBool( KeyedUnarchiverRef ref, CFStringRef key ) = Boolean
KeyedUnarchiverDecodeBytes( KeyedUnarchiverRef ref, CFStringRef key, NSUInteger *returnedLength ) = ptr
KeyedUnarchiverDecodeDouble( KeyedUnarchiverRef ref, CFStringRef key ) = double
KeyedUnarchiverDecodeFloat( KeyedUnarchiverRef ref, CFStringRef key ) = float
KeyedUnarchiverDecodeInt32( KeyedUnarchiverRef ref, CFStringRef key ) = SInt32
KeyedUnarchiverDecodeInt64( KeyedUnarchiverRef ref, CFStringRef key ) = SInt64
KeyedUnarchiverDecodeObject( KeyedUnarchiverRef ref, CFStringRef key ) = CTypeRef
KeyedUnarchiverFinishDecoding( KeyedUnarchiverRef ref )
```

Classes

```
KeyedUnarchiverSetClassForClassName( ClassRef class, CFStringRef codedName )// class method
KeyedUnarchiverClassForClassName( CFStringRef codedName ) = ClassRef// class method
KeyedUnarchiverSetClass( KeyedArchiverRef ref, ClassRef class, CFStringRef codedName )// instance method
KeyedUnarchiverClass( KeyedArchiverRef ref, CFStringRef codedName ) = ClassRef// instance method
```

Initializers

```
KeyedUnarchiverForReadingFromData( CFDataRef dta, ErrorRef *err ) = KeyedUnarchiverRef// macOS 10.13+
```

Instance properties

```
KeyedUnarchiverDecodingFailurePolicy( KeyedArchiverRef ref ) = NSDecodingFailurePolicy// macOS 10.12+ - Apple docs say macOS 10.11+ but clang throws up warning
KeyedUnarchiverSetDecodingFailurePolicy( KeyedArchiverRef ref, NSDecodingFailurePolicy policy )// macOS 10.12+ - Apple docs say macOS 10.11+ but clang throws up warning
```

Type methods

```
KeyedUnarchiverUnarchivedObjectOfClass( ClassRef class, CFDataRef fromData, ErrorRef *err ) = CTypeRef// macOS 10.14+ - Apple docs say macOS 10.13+ but clang throws up warning
KeyedUnarchiverUnarchivedObjectOfClasses( CFSetRef classes, CFDataRef fromData, ErrorRef *err ) = CTypeRef// macOS 10.14+ - Apple docs say macOS 10.13+ but clang throws up warning
```

### See Also

[KeyedArchiver](#)

### Apple documentation

[NSKeyedUnarchiver](#)



## LayoutManager

### Functions

#### Init

```
LayoutManagerInit = LayoutManagerRef// autoreleased
```

#### Text storage

```
LayoutManagerTextStorage( LayoutManagerRef ref ) = TextStorageRef  
LayoutManagerReplaceTextStorage( LayoutManagerRef ref, TextStorageRef ts )
```

#### Text containers

```
LayoutManagerTextContainers( LayoutManagerRef ref ) = CFArrayRef  
LayoutManagerAddTextContainer( LayoutManagerRef ref, TextContainerRef tc )  
LayoutManagerInsertTextContainer( LayoutManagerRef ref, TextContainerRef tc, NSUInteger index )  
LayoutManagerRemoveTextContainer( LayoutManagerRef ref, NSUInteger index )  
LayoutManagerSetTextContainer( LayoutManagerRef ref, TextContainerRef tc, CFRange glyphRange )  
LayoutManagerTextContainerChangedGeometry( LayoutManagerRef ref, TextContainerRef tc )  
LayoutManagerTextContainerChangedTextView( LayoutManagerRef ref, TextContainerRef tc )  
LayoutManagerTextContainerForGlyph( LayoutManagerRef ref, NSUInteger index, CFRange *effectiveRange ) =  
TextContainerRef  
LayoutManagerTextContainerForGlyphWithoutAdditionalLayout( LayoutManagerRef ref, NSUInteger index, CFRange  
*effectiveRange, Boolean flag ) = TextContainerRef  
LayoutManagerUsedRectForTextContainer( LayoutManagerRef ref, TextContainerRef tc ) = CGRect
```

#### Global layout manager options

```
LayoutManagerAllowsNonContiguousLayout( LayoutManagerRef ref ) = Boolean  
LayoutManagerSetAllowsNonContiguousLayout( LayoutManagerRef ref, Boolean flag )  
LayoutManagerHasNonContiguousLayout( LayoutManagerRef ref ) = Boolean  
LayoutManagerHyphenationFactor( LayoutManagerRef ref ) = CGFloat  
LayoutManagerSetHyphenationFactor( LayoutManagerRef ref, CGFloat factor )  
LayoutManagerShowsInvisibleCharacters( LayoutManagerRef ref ) = Boolean  
LayoutManagerSetShowsInvisibleCharacters( LayoutManagerRef ref, Boolean flag )  
LayoutManagerShowsControlCharacters( LayoutManagerRef ref ) = Boolean  
LayoutManagerSetShowsControlCharacters( LayoutManagerRef ref, Boolean flag )  
LayoutManagerUsesFontLeading( LayoutManagerRef ref ) = Boolean  
LayoutManagerSetUsesFontLeading( LayoutManagerRef ref, Boolean flag )  
LayoutManagerBackgroundLayoutEnabled( LayoutManagerRef ref ) = Boolean  
LayoutManagerSetBackgroundLayoutEnabled( LayoutManagerRef ref, Boolean flag )
```

#### Invalidate glyphs and layout

```
LayoutManagerInvalidateDisplayForCharacterRange( LayoutManagerRef ref, CFRange range )  
LayoutManagerInvalidateDisplayForGlyphRange( LayoutManagerRef ref, CFRange range )  
LayoutManagerInvalidateGlyphsForCharacterRange( LayoutManagerRef ref, CFRange range, NSInteger delta, CFRange  
*actualCharacterRange )  
LayoutManagerInvalidateLayoutForCharacterRange( LayoutManagerRef ref, CFRange range, CFRange *actualCharacterRange )  
LayoutManagerProcessEditingForTextStorage( LayoutManagerRef ref, TextStorageRef textStorage,  
NSTextStorageEditActions edited, CFRange range, NSInteger delta, CFRange invalidatedRange )// macOS 10.11+
```

#### Glyph generation and layout

```
LayoutManagerEnsureGlyphsForCharacterRange( LayoutManagerRef ref, CFRange range )  
LayoutManagerEnsureGlyphsForGlyphRange( LayoutManagerRef ref, CFRange range )  
LayoutManagerEnsureLayoutForBoundingRect( LayoutManagerRef ref, CGRect r, TextContainerRef tc )  
LayoutManagerEnsureLayoutForCharacterRange( LayoutManagerRef ref, CFRange range )  
LayoutManagerEnsureLayoutForGlyphRange( LayoutManagerRef ref, CFRange range )  
LayoutManagerEnsureLayoutForTextContainer( LayoutManagerRef ref, TextContainerRef tc )  
LayoutManagerGlyphGenerator( LayoutManagerRef ref ) = GlyphGeneratorRef
```

#### Accessing glyphs

```
LayoutManagerGetGlyphs( LayoutManagerRef ref, CFRange range, CGGlyph *glyphBuffer, NSGlyphProperty *props,  
NSUInteger *charIndexBuffer, unsigned char *bidiLevelsBuffer ) = NSUInteger// macOS 10.11+  
LayoutManagerCGGlyphAtIndex( LayoutManagerRef ref, NSUInteger glyphIndex ) = CGGlyph// macOS 10.11+  
LayoutManagerCGGlyphAtIndexIsValidIndex( LayoutManagerRef ref, NSUInteger glyphIndex, Boolean *isValidIndex ) =  
CGGlyph// macOS 10.11+  
LayoutManagerSetCGGlyphs( LayoutManagerRef ref, CGGlyph *glyphs, NSGlyphProperty *props, NSUInteger *charIndexes,  
FontRef font, CFRange glyphRange )// macOS 10.11+  
LayoutManagerCharacterIndexForGlyphAtIndex( LayoutManagerRef ref, NSUInteger glyphIndex ) = NSUInteger  
LayoutManagerGlyphIndexForCharacterAtIndex( LayoutManagerRef ref, NSUInteger charIndex ) = NSUInteger  
LayoutManagerIsValidGlyphIndex( LayoutManagerRef ref, NSUInteger glyphIndex ) = Boolean  
LayoutManagerNumberOfGlyphs( LayoutManagerRef ref ) = NSUInteger  
LayoutManagerPropertyForGlyphAtIndex( LayoutManagerRef ref, NSUInteger glyphIndex ) = NSGlyphProperty// macOS 10.11+
```

#### Set layout info

```
LayoutManagerSetAttachmentSize( LayoutManagerRef ref, CGSize size, CFRange glyphRange )
```

```
LayoutManagerSetDrawsOutsideLineFragment( LayoutManagerRef ref, Boolean flag, NSUInteger glyphIndex )
LayoutManagerSetExtraLineFragmentRect( LayoutManagerRef ref, CGRect r, CGRect usedRect, TextContainerRef
textContainer )
LayoutManagerSetLineFragmentRect( LayoutManagerRef ref, CGRect r, CFRange glyphRange, CGRect usedRect )
LayoutManagerSetLocation( LayoutManagerRef ref, CGPoint pt, CFRange startOfGlyphRange )
LayoutManagerSetNotShownAttribute( LayoutManagerRef ref, Boolean flag, NSUInteger glyphIndex )
```

#### Get layout info

```
LayoutManagerAttachmentSizeForGlyphAtIndex( LayoutManagerRef ref, NSUInteger index ) = CGSize
LayoutManagerDrawsOutsideLineFragmentForGlyphAtIndex( LayoutManagerRef ref, NSUInteger index ) = Boolean
LayoutManagerExtraLineFragmentRect( LayoutManagerRef ref ) = CGRect
LayoutManagerExtraLineFragmentTextContainer( LayoutManagerRef ref ) = TextContainerRef
LayoutManagerExtraLineFragmentUsedRect( LayoutManagerRef ref ) = CGRect
LayoutManagerFirstUnlaidCharacterIndex( LayoutManagerRef ref ) = NSUInteger
LayoutManagerFirstUnlaidGlyphIndex( LayoutManagerRef ref ) = NSUInteger
LayoutManagerGetFirstUnlaidCharacterIndex( LayoutManagerRef ref, NSUInteger *charIndex, NSUInteger *glyphIndex )
LayoutManagerLineFragmentRectForGlyphAtIndex( LayoutManagerRef ref, NSUInteger index, CFRange *effectiveRange ) =
CGRect
LayoutManagerLineFragmentRectForGlyphAtIndexWithoutAdditionalLayout( LayoutManagerRef ref, NSUInteger index, CFRange
*effectiveRange, Boolean flag ) = CGRect
LayoutManagerExtraLineFragmentUsedRectForGlyphAtIndex( LayoutManagerRef ref, NSUInteger index, CFRange
*effectiveRange ) = CGRect
LayoutManagerExtraLineFragmentUsedRectForGlyphAtIndexWithoutAdditionalLayout( LayoutManagerRef ref, NSUInteger
index, CFRange *effectiveRange, Boolean flag ) = CGRect
LayoutManagerLocationForGlyphAtIndex( LayoutManagerRef ref, NSUInteger index ) = CGPoint
LayoutManagerNotShownAttributeForGlyphAtIndex( LayoutManagerRef ref, NSUInteger index ) = Boolean
LayoutManagerTruncatedGlyphRangeInLineFragmentForGlyphAtIndex( LayoutManagerRef ref, NSUInteger index ) = CFRange//
macOS 10.11+
```

#### Advanced layout queries

```
LayoutManagerBoundingRectForGlyphRange( LayoutManagerRef ref, CFRange glyphRange, TextContainerRef tc ) = CGRect
LayoutManagerCharacterIndexForPoint( LayoutManagerRef ref, CGPoint pt, TextContainerRef tc, CGFloat *partialFraction
) = NSUInteger
LayoutManagerCharacterRangeForGlyphRange( LayoutManagerRef ref, CFRange glyphRange, CFRange *actualGlyphRange ) =
CFRange
```

```
LayoutManagerGlyphRangeForBoundingRect( LayoutManagerRef ref, CGRect bounds, TextContainerRef tc ) = CFRange
```

## Apple documentation

[NSLayoutManager](#)



## LevelIndicator

statement

### Syntax

**levelindicator** *tag, enabled, value, rect, min, max, warning, critical, tickmarks, majorTickMarks, wndTag*

### Description

The **levelindicator** statement puts a new levelindicator in the current output cocoa window, or alters an existing levelindicator's characteristics.

### Parameters

<i>tag</i>	A number to identify the levelindicator. A negative tag hides the levelindicator.
<i>enabled</i>	Enable or disable the levelindicator.
<i>value</i>	The levelindicator's value.
<i>rect</i>	Origin and size of the levelindicator in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>min</i>	The levelindicator's minimum value (default = 0).
<i>max</i>	The levelindicator's maximum value (default = 2).
<i>warning</i>	The levelindicator's warning value (default = 0).
<i>critical</i>	The levelindicator's critical value (default = 0).
<i>tickMarks</i>	The number of tick marks (default = 0).
<i>majorTickMarks</i>	The number of major tick marks must be less than or equal to the number of tick marks (default = 0).
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

[\\_btnClick](#)

### Functions

```
LevelIndicatorWithTag( NSInteger tag ) = LevelIndicatorRef
LevelIndicatorExists( NSInteger tag ) = Boolean
```

Init

```
LevelIndicatorInit( NSInteger tag, CGRect r ) = LevelIndicatorRef// autoreleased
```

Range

```
LevelIndicatorMinValue( NSInteger tag ) = double
LevelIndicatorSetMinValue( NSInteger tag, double value )
LevelIndicatorMaxValue( NSInteger tag ) = double
LevelIndicatorSetMaxValue( NSInteger tag, double value )
LevelIndicatorWarningValue( NSInteger tag ) = double
LevelIndicatorSetWarningValue( NSInteger tag, double value )
LevelIndicatorCriticalValue( NSInteger tag ) = double
LevelIndicatorSetCriticalValue( NSInteger tag, double value )
```

Tick marks and style

```
LevelIndicatorTickMarkPosition( NSInteger tag ) = NSTickMarkPosition
LevelIndicatorSetTickMarkPosition( NSInteger tag, NSTickMarkPosition position )
LevelIndicatorNumberOfTickMarks( NSInteger tag ) = NSInteger
LevelIndicatorSetNumberOfTickMarks( NSInteger tag, NSInteger tickMarks )
LevelIndicatorNumberOfMajorTickMarks( NSInteger tag ) = NSInteger
LevelIndicatorSetNumberOfMajorTickMarks( NSInteger tag, NSInteger tickMarks )
LevelIndicatorRectOfTickMarkAtIndex( NSInteger tag, NSInteger index ) = CGRect
LevelIndicatorStyle( NSInteger tag ) = NSLevelIndicatorStyle// macOS 10.10 or higher
LevelIndicatorSetStyle( NSInteger tag, NSLevelIndicatorStyle style )// macOS 10.10 or higher
```

#### Instance properties

```
LevelIndicatorIsEditable( NSInteger tag ) = Boolean
LevelIndicatorSetEditable( NSInteger tag, Boolean flag )
LevelIndicatorImage( NSInteger tag ) = ImageRef
LevelIndicatorSetImage( NSInteger tag, ImageRef image )
LevelIndicatorSetImageNamed( NSInteger tag, CFStringRef imageName )
```

• note: the following calls require BOTH 'Min deployment' and 'Base SDK' to be set at 10.13 or higher •

```
LevelIndicatorCriticalFillColor( NSInteger tag ) = ColorRef
LevelIndicatorSetCriticalFillColor( NSInteger tag, ColorRef col )
LevelIndicatorDrawsTieredCapacityLevels( NSInteger tag ) = Boolean
LevelIndicatorSetDrawsTieredCapacityLevels( NSInteger tag, Boolean flag )
LevelIndicatorFillColor( NSInteger tag ) = ColorRef
LevelIndicatorSetFillColor( NSInteger tag, ColorRef col )
LevelIndicatorPlaceholderVisibility( NSInteger tag ) = NSLevelIndicatorPlaceholderVisibility
LevelIndicatorSetPlaceholderVisibility( NSInteger tag, NSLevelIndicatorPlaceholderVisibility visibility )
LevelIndicatorRatingImage( NSInteger tag ) = ImageRef
LevelIndicatorSetRatingImage( NSInteger tag, ImageRef image )
LevelIndicatorRatingPlaceholderImage( NSInteger tag ) = ImageRef
LevelIndicatorSetRatingPlaceholderImage( NSInteger tag, ImageRef image )
LevelIndicatorSetRatingPlaceholderImageNamed( NSInteger tag, CFStringRef imageName )
LevelIndicatorWarningFillColor( NSInteger tag ) = ColorRef
LevelIndicatorSetWarningFillColor( NSInteger tag, ColorRef col )
```





#### See Also





[Control](#), [View](#)

#### Apple documentation

[NSLevelIndicator](#)



Level Indicator Styles (macOS 10.12 or earlier)		Recommended Heights		
		Standard	Minor Ticks	Major Ticks
	NSDiscreteCapacityLevelIndicatorStyle	18	22	25
	NSContinuousCapacityLevelIndicatorStyle	16	20	23
	NSRatingLevelIndicatorStyle	13	17	20
	NSRelevancyLevelIndicatorStyle	12	16	19

Level Indicator Styles (macOS 10.13+)		Recommended Heights	
		Standard	Ticks Marks
	NSDiscreteCapacityLevelIndicatorStyle	16	23
	NSContinuousCapacityLevelIndicatorStyle	16	23
	NSRatingLevelIndicatorStyle	12	19
	NSRelevancyLevelIndicatorStyle	12	19



## LinguisticTagger

### Functions

#### Init

```
LinguisticTaggerInit( CFArrayRef tagSchemes, NSUInteger options ) = LinguisticTaggerRef// autoreleased
```

#### String

```
LinguisticTaggerString( LinguisticTaggerRef ref ) = CFStringRef  
LinguisticTaggerSetString( LinguisticTaggerRef ref, CFStringRef string )
```

#### Tag schemes

```
LinguisticTaggerAvailableTagSchemesForUnit( NSLinguisticTaggerUnit unit, CFStringRef language ) = CFArrayRef// macOS 10.13+  
LinguisticTaggerAvailableTagSchemesForLanguage( CFStringRef language ) = CFArrayRef  
LinguisticTaggerTagSchemes( LinguisticTaggerRef ref ) = CFArrayRef
```

#### Determine dominant language and orthography

```
LinguisticTaggerDominantLanguageForString( CFStringRef string ) = CFStringRef// macOS 10.13+  
LinguisticTaggerDominantLanguage( LinguisticTaggerRef ref ) = CFStringRef// macOS 10.13+  
LinguisticTaggerOrthographyAtIndex( LinguisticTaggerRef ref, NSUInteger index, CFRange *effectiveRange ) = OrthographyRef  
LinguisticTaggerSetOrthography( LinguisticTaggerRef ref, OrthographyRef orthRef, CFRange range )
```

#### Enumerate

```
LinguisticTaggerEnumerateTagsForUnit( LinguisticTaggerRef ref, CFRange range, NSLinguisticTaggerUnit unit, CFStringRef scheme, NSLinguisticTaggerOptions options, ptr callback, ptr userData )// macOS 10.13+  
LinguisticTaggerEnumerateTagsForScheme( LinguisticTaggerRef ref, CFRange range, CFStringRef scheme, NSLinguisticTaggerOptions options, ptr callback, ptr userData )  
LinguisticTaggerEnumerateTagsForString( CFStringRef string, CFRange range, NSLinguisticTaggerUnit unit, CFStringRef scheme, NSLinguisticTaggerOptions options, OrthographyRef orthRef, ptr callback, ptr userData )// macOS 10.13+
```

#### Get

```
LinguisticTaggerTagForUnit( LinguisticTaggerRef ref, NSUInteger index, NSLinguisticTaggerUnit unit, CFStringRef scheme, CFRange *tokenRange ) = CFStringRef// macOS 10.13+  
LinguisticTaggerTagForScheme( LinguisticTaggerRef ref, NSUInteger index, CFStringRef scheme, CFRange *tokenRange, CFRange *sentenceRange ) = CFStringRef  
LinguisticTaggerTagForString( CFStringRef string, NSUInteger index, NSLinguisticTaggerUnit unit, CFStringRef scheme, OrthographyRef orthRef, CFRange *tokenRange ) = CFStringRef// macOS 10.13+  
LinguisticTaggerTagsForUnit( LinguisticTaggerRef ref, CFRange range, NSLinguisticTaggerUnit unit, CFStringRef scheme, NSLinguisticTaggerOptions options, CFArrayRef *tokenRanges ) = CFArrayRef// macOS 10.13+  
LinguisticTaggerTagsForScheme( LinguisticTaggerRef ref, CFRange range, CFStringRef scheme, NSLinguisticTaggerOptions options, CFArrayRef *tokenRanges ) = CFArrayRef  
LinguisticTaggerTagsForString( CFStringRef string, CFRange range, NSLinguisticTaggerUnit unit, CFStringRef scheme, NSLinguisticTaggerOptions options, OrthographyRef orthRef, CFArrayRef *tokenRanges ) = CFArrayRef// macOS 10.13+
```

#### Range of unit token

```
LinguisticTaggerTokenRangeAtIndex( LinguisticTaggerRef ref, NSUInteger index, NSLinguisticTaggerUnit unit ) = CFRange// macOS 10.13+  
LinguisticTaggerSentenceRangeForRange( LinguisticTaggerRef ref, CFRange range ) = CFRange
```

#### Possible tags

```
LinguisticTaggerPossibleTagsAtIndex( LinguisticTaggerRef ref, NSUInteger index, CFStringRef scheme, CFRange *tokenRange, CFRange *sentenceRange, CFArrayRef *scores ) = CFArrayRef
```

#### Notify changes to analyzed string

```
LinguisticTaggerStringEditedInRange( LinguisticTaggerRef ref, CFRange newRange, NSInteger delta )
```

### Apple documentation

[NSLinguisticTagger](#)



## Locale

### Functions

Init  
`LocaleWithIdentifier( CFStringRef identifier ) = CFLocaleRef`

User's locale  
`LocaleAutoupdatingCurrent = CFLocaleRef`  
`LocaleCurrent = CFLocaleRef`  
`LocaleSystem = CFLocaleRef`

Known identifiers and codes  
`LocaleAvailableIdentifiers = CFArrayRef`  
`LocaleISOCountryCodes = CFArrayRef`  
`LocaleISOLanguageCodes = CFArrayRef`  
`LocaleISOCurrencyCodes = CFArrayRef`  
`LocaleCommonISOCurrencyCodes = CFArrayRef`

Converting between identifiers  
`LocaleCanonicalIdentifierFromString( CFStringRef string ) = CFStringRef`  
`LocaleComponentsFromIdentifier( CFStringRef identifier ) = CFDictionaryRef`  
`LocaleIdentifierFromComponents( CFDictionaryRef components ) = CFStringRef`  
`LocaleCanonicalLanguageIdentifierFromString( CFStringRef string ) = CFStringRef`  
`LocaleIdentifierFromWindowsLocaleCode( UInt32 lcid ) = CFStringRef`  
`LocaleWindowsLocaleCodeFromIdentifier( CFStringRef identifier ) = UInt32`

Locale info  
`LocaleIdentifier( CFLocaleRef locale ) = CFStringRef`  
`LocaleCountryCode( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleLanguageCode( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleScriptCode( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleVariantCode( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleExemplarCharacterSet( CFLocaleRef locale ) = CFCharacterSetRef// macOS 10.12+`  
`LocaleCollationIdentifier( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleCollatorIdentifier( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleUsesMetricSystem( CFLocaleRef locale ) = Boolean// macOS 10.12+`  
`LocaleDecimalSeparator( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleGroupingSeparator( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleCurrencyCode( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleCurrencySymbol( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleCalendarIdentifier( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleQuotationBeginDelimiter( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleQuotationEndDelimiter( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleAlternateQuotationBeginDelimiter( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`  
`LocaleAlternateQuotationEndDelimiter( CFLocaleRef locale ) = CFStringRef// macOS 10.12+`

Display info  
`LocaleLocalizedStringForIdentifier( CFLocaleRef locale, CFStringRef identifier ) = CFStringRef// macOS 10.12+`  
`LocaleLocalizedStringForCountryCode( CFLocaleRef locale, CFStringRef code ) = CFStringRef// macOS 10.12+`  
`LocaleLocalizedStringForLanguageCode( CFLocaleRef locale, CFStringRef code ) = CFStringRef// macOS 10.12+`  
`LocaleLocalizedStringForScriptCode( CFLocaleRef locale, CFStringRef code ) = CFStringRef// macOS 10.12+`  
`LocaleLocalizedStringForVariantCode( CFLocaleRef locale, CFStringRef code ) = CFStringRef// macOS 10.12+`  
`LocaleLocalizedStringForCollationIdentifier( CFLocaleRef locale, CFStringRef identifier ) = CFStringRef// macOS 10.12+`  
`LocaleLocalizedStringForCollatorIdentifier( CFLocaleRef locale, CFStringRef identifier ) = CFStringRef// macOS 10.12+`  
`LocaleLocalizedStringForCurrencyCode( CFLocaleRef locale, CFStringRef code ) = CFStringRef// macOS 10.12+`  
`LocaleLocalizedStringForCalendarIdentifier( CFLocaleRef locale, CFStringRef identifier ) = CFStringRef// macOS 10.12+`

Info by key  
`LocaleObjectForKey( CFLocaleRef locale, CFStringRef key ) = CFTypeRef`  
`LocaleDisplayNameForKey( CFLocaleRef locale, CFStringRef key, CFTypeRef value ) = CFStringRef`

Preferred languages  
`LocalePreferredLanguages = CFArrayRef`

Line and character direction  
`LocaleCharacterDirectionForLanguage( CFStringRef isoLangCode ) = NSLocaleLanguageDirection`  
`LocaleLineDirectionForLanguage( CFStringRef isoLangCode ) = NSLocaleLanguageDirection`





---

## MagnificationGestureRecognizer

### Functions

Init

```
MagnificationGestureRecognizerInit( ptr callback, ptr userData ) = MagnificationGestureRecognizerRef// autoreleased
```

Magnification factor

```
MagnificationGestureRecognizerMagnification( MagnificationGestureRecognizerRef ref ) = CGFloat
```

```
MagnificationGestureRecognizerSetMagnification( MagnificationGestureRecognizerRef ref, CGFloat rotation )
```

### Apple documentation

[NSLocale](#)



## MKMapView

statement

### Syntax

**mkmapview** *tag*, *rect*, *wndTag*

### Description

The **mkmapview** statement puts a new mkmapview in the current output cocoa window, or alters an existing mkmapview's characteristics.

### Parameters

<i>tag</i>	A number to identify the mapview. A negative tag hides the view.
<i>rect</i>	Origin and size of the view in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

In order to receive mapview dialog events, the mapview delegate must be installed with [MKMapViewInstallDelegate](#).

```
_mapViewRegionWillChange
_mapViewDidChangeVisibleRegion// macOS 10.13+
_mapViewRegionDidChange
_mapViewWillStartLoadingMap
_mapViewDidFinishLoadingMap
_mapViewDidFailLoadingMap
_mapViewWillStartRenderingMap
_mapViewDidFinishRenderingMap
_mapViewWillStartLocatingUser
_mapViewDidStopLocatingUser
_mapViewDidUpdateUserLocation
_mapViewDidFailToLocateUser
_mapViewViewForAnnotation
_mapViewDidAddAnnotationViews
_mapViewClusterAnnotationForMemberAnnotations// macOS 10.13+
_mapViewAnnotationViewDidChangeDragState
_mapViewDidSelectAnnotationView
_mapViewDidDeselectAnnotationView
_mapViewRendererForOverlay
_mapViewDidAddOverlayRenderers
```

### Functions

```
MKMapViewWithTag( NSInteger tag ) = MKMapViewRef
WKMapViewExists( NSInteger tag ) = Boolean
```

### Properties

```
MKMapViewMapType( NSInteger tag ) = MKMapType
MKMapViewSetMapType( NSInteger tag, MKMapType type )
MKMapViewIsZoomEnabled( NSInteger tag ) = Boolean
MKMapViewSetZoomEnabled( NSInteger tag, Boolean flag )
MKMapViewIsScrollEnabled( NSInteger tag ) = Boolean
MKMapViewSetScrollEnabled( NSInteger tag, Boolean flag )
MKMapViewIsPitchEnabled( NSInteger tag ) = Boolean
MKMapViewSetPitchEnabled( NSInteger tag, Boolean flag )
MKMapViewIsRotateEnabled( NSInteger tag ) = Boolean
MKMapViewSetRotateEnabled( NSInteger tag, Boolean flag )
```

### Visible portion

```
MKMapViewRegion( NSInteger tag ) = MKCoordinateRegion
MKMapViewSetRegion( NSInteger tag, MKCoordinateRegion region, Boolean animated )
MKMapViewCenterCoordinate( NSInteger tag ) = CLLocationCoordinate2D
MKMapViewSetCenterCoordinate( NSInteger tag, CLLocationCoordinate2D coordinate, Boolean animated )
MKMapViewShowAnnotations( NSInteger tag, CFArrayRef annotations, Boolean animated )
MKMapViewVisibleMapRect( NSInteger tag ) = MKMapRect
MKMapViewSetVisibleMapRect( NSInteger tag, MKMapRect rect, Boolean animated )
MKMapViewSetVisibleMapRectEdgePadding( NSInteger tag, MKMapRect rect, NSEdgeInsets insets, Boolean animated )
```

## Appearance

```
MKMapViewCamera( NSInteger tag ) = MKMapCameraRef
MKMapViewSetCamera( NSInteger tag, MKMapCameraRef camera, Boolean animated )
MKMapViewShowsPointsOfInterest( NSInteger tag ) = Boolean
MKMapViewSetShowsPointsOfInterest( NSInteger tag, Boolean flag )
MKMapViewShowsBuildings( NSInteger tag ) = Boolean
MKMapViewSetShowsBuildings( NSInteger tag, Boolean flag )
MKMapViewShowsCompass( NSInteger tag ) = Boolean
MKMapViewSetShowsCompass( NSInteger tag, Boolean flag )
MKMapViewShowsZoomControls( NSInteger tag ) = Boolean
MKMapViewSetShowsZoomControls( NSInteger tag, Boolean flag )
MKMapViewShowsScale( NSInteger tag ) = Boolean// macOS 10.10+
MKMapViewSetShowsScale( NSInteger tag, Boolean flag )// macOS 10.10+
MKMapViewShowsTraffic( NSInteger tag ) = Boolean
MKMapViewSetShowsTraffic( NSInteger tag, Boolean flag )
```

## User location

```
MKMapViewShowsUserLocation( NSInteger tag ) = Boolean
MKMapViewSetShowsUserLocation( NSInteger tag, Boolean flag )
MKMapViewIsUserLocationVisible( NSInteger tag ) = Boolean
MKMapViewUserLocation( NSInteger tag ) = MKUserLocationRef
```

## Annotating

```
MKMapViewAnnotations( NSInteger tag ) = CFArrayRef
MKMapViewAddAnnotation( NSInteger tag, MKAnnotationRef annotation )
MKMapViewAddAnnotations( NSInteger tag, CFArrayRef annotations )
MKMapViewRemoveAnnotation( NSInteger tag, MKAnnotationRef annotation )
MKMapViewRemoveAnnotations( NSInteger tag, CFArrayRef annotations )
MKMapViewAnnotationsInMapRect( NSInteger tag, MKMapRect rect ) = CFSetRef
```

## Annotation selections

```
MKMapViewAnnotationVisibleRect( NSInteger tag ) = CGRect
MKMapViewSelectedAnnotations( NSInteger tag ) = CFArrayRef
MKMapViewSelectAnnotation( NSInteger tag, MKAnnotationRef annotation, Boolean animated )
MKMapViewDeselectAnnotation( NSInteger tag, MKAnnotationRef annotation, Boolean animated )
```

## Annotation views

```
MKMapViewDequeReusableAnnotationViewForAnnotation( NSInteger tag, CFStringRef identifier, MKAnnotationRef
annotation ) = MKAnnotationViewRef// macOS 10.13+
MKMapViewDequeueReusableAnnotationView( NSInteger tag, CFStringRef identifier ) = MKAnnotationViewRef
MKMapViewViewForAnnotation( NSInteger tag, MKAnnotationRef annotation ) = MKAnnotationViewRef
```

## Accessing overlays

```
MKMapViewOverlays( NSInteger tag ) = CFArrayRef
MKMapViewOverlaysInLevel( NSInteger tag, MKOverlayLevel level ) = CFArrayRef
MKMapViewRendererForOverlay( NSInteger tag, MKOverlayRef overlay ) = MKOverlayRendererRef
```

## Add overlays

```
MKMapViewAddOverlayAtLevel( NSInteger tag, MKOverlayRef overlay, MKOverlayLevel level )
MKMapViewAddOverlaysAtLevel( NSInteger tag, CFArrayRef overlays, MKOverlayLevel level )
MKMapViewAddOverlay( NSInteger tag, MKOverlayRef overlay )
MKMapViewAddOverlays( NSInteger tag, CFArrayRef overlays )
MKMapViewInsertOverlayAtLevel( NSInteger tag, MKOverlayRef overlay, NSUInteger index, MKOverlayLevel level )
MKMapViewInsertOverlay( NSInteger tag, MKOverlayRef overlay, NSUInteger index )
MKMapViewInsertOverlayAboveOverlay( NSInteger tag, MKOverlayRef overlay1, MKOverlayRef overlay2 )
MKMapViewInsertOverlayBelowOverlay( NSInteger tag, MKOverlayRef overlay1, MKOverlayRef overlay2 )
MKMapViewExchangeOverlay( NSInteger tag, MKOverlayRef overlay1, MKOverlayRef overlay2 )
MKMapViewExchangeOverlayAtIndex( NSInteger tag, NSUInteger index1, NSUInteger index2 )
```

## Remove overlays

```
MKMapViewRemoveOverlay( NSInteger tag, MKOverlayRef overlay )
MKMapViewRemoveOverlays( NSInteger tag, CFArrayRef overlays )
```

## Regions and rectangles

```
MKMapViewRegionThatFits( NSInteger tag, MKCoordinateRegion region ) = MKCoordinateRegion
MKMapViewMapRectThatFits( NSInteger tag, MKMapRect rect ) = MKMapRect
MKMapViewMapRectThatFitsEdgePadding( NSInteger tag, MKMapRect rect, UIEdgeInsets insets ) = MKMapRect
```

## Custom

```
MKMapViewInstallDelegate( NSInteger tag )
```

## Apple documentation

[MKMapView](#)



---

## MenuEvent

### Syntax

```
menuIndex = menu(_menuID)  
itemIndex = menu(_itemID)
```

### Description

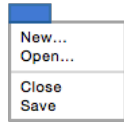
This supplements FBHelp's **menu** function documentation.

Cocoa menus and menu items are handled automatically and intercepted with the traditional 'on menu fn xxxx'. Note that cocoa menu items are zero-based and cocoa menus require [CocoaInit](#).

### Functions

```
MenuEventSetBool( Boolean value )
```





## Cocoa Menu

statement

### Syntax

**cocoa menu** *menuIndex*, *itemIndex*, *enabled*, *title*, *keyEquivalent*, *keyEquivalentModifier*

### Description

Use this statement to:

- Create a new cocoa menu or menu item;
- Alter the title or enabled state of an existing cocoa menu or menu item.

### Parameters

<i>menuIndex</i>	The index of the menu. Menu indexes 0-100 are reserved for menu bar menus. A menu with an index of 101 or higher is not inserted in the menu bar but stored for later use (hierarchical menus, popupbutton menus, etc).
<i>itemIndex</i>	When creating or altering the menu itself, omit this parameter. Otherwise it's the menu item index.
<i>enabled</i>	Enable or disable a menu item.
<i>title</i>	The menu or item title.
<i>keyEquivalent</i>	The key equivalent as a string.
<i>keyEquivalentModifier</i>	Modifier masks can be combined using '+'. <a href="#">NSAlphaShiftKeyMask</a> <a href="#">NSShiftKeyMask</a> <a href="#">NSControlKeyMask</a> <a href="#">NSAlternateKeyMask</a> <a href="#">NSCommandKeyMask</a> <a href="#">NSNumericPadKeyMask</a> <a href="#">NSHelpKeyMask</a> <a href="#">NSFunctionKeyMask</a>

### Menu Events

```
menuItemIndex = menu(_menuItemID)
itemIndex = menu(_itemID)
```

### Functions

```
MenuAtIndex( NSInteger index ) = CocoaMenuRef
MenuExists( NSInteger index ) = Boolean
MenuIndex( CocoaMenuRef m ) = NSInteger
```

```
Menu bar
MenuBarVisible = Boolean
MenuBarSetVisible( Boolean flag )
MenuBarHeight = CGFloat
```

```
Init
MenuWithTitle( CFStringRef title ) = CocoaMenuRef
```

Adding and removing

```
MenuInsertItem( NSInteger menuIndex, NSInteger itemIndex, CFStringRef title, CFStringRef keyEquivalent,
NSEventModifierFlags keyEquivalentModifierMask )
MenuAddItem( NSInteger menuIndex, CFStringRef title, CFStringRef keyEquivalent, NSEventModifierFlags
keyEquivalentModifierMask )
MenuRemoveItem( NSInteger menuIndex, NSInteger itemIndex )
MenuRemoveAllItems( NSInteger menuIndex )
```

Finding items

```
MenuItemWithTitle( NSInteger menuIndex, CFStringRef title ) = NSInteger
MenuNumberOfItems( NSInteger menuIndex ) = NSInteger
MenuItemArray( NSInteger menuIndex ) = CFArrayRef
```

Find indices of items

```
MenuIndexofItemWithTitle( NSInteger menuIndex, CFStringRef title ) = NSInteger
MenuIndexofItemWithTag( NSInteger menuIndex, NSInteger tag ) = NSInteger
```

#### Submenus

```
MenuSetSubmenu( NSInteger parMenuItemIndex, NSInteger parItemIndex, NSInteger submenuIndex )
```

#### Enable and disable

```
MenuSetAutoenablesItems( NSInteger menuIndex, Boolean flag )
```

#### Font

```
MenuFont( NSInteger menuIndex ) = CTNSFontRef  
MenuSetFont( NSInteger menuIndex, CTNSFontRef font )  
MenuSetFontWithName( NSInteger tag, CFStringRef name, CGFloat size )
```

#### Title

```
MenuTitle( NSInteger menuIndex ) = CFStringRef
```

#### Size

```
MenuMinimumWidth( NSInteger menuIndex ) = CGFloat  
MenuSetMinimumWidth( NSInteger menuIndex, CGFloat width )  
MenuSize( NSInteger menuIndex ) = CGSize
```

#### Context-sensitive help

```
MenuPopUp( NSInteger menuIndex, NSInteger itemIndex, CGPoint location, NSInteger viewTag ) = Boolean
```

#### State column

```
MenuShowsStateColumn( NSInteger menuIndex ) = Boolean  
MenuSetShowsStateColumn( NSInteger menuIndex, Boolean flag )
```

#### Highlighting

```
MenuHighlightedItem( NSInteger menuIndex ) = NSInteger
```

#### Custom

```
MenuSetTag( NSInteger menuIndex, NSInteger itemIndex, NSInteger tag )  
MenuSetValidateItemsCallback( ptr callback )  
MenuSetOneItemOnState( NSInteger menuID, NSInteger itemID )  
MenuIsHidden( NSInteger menuIndex ) = Boolean  
MenuSetHidden( NSInteger menuIndex, Boolean flag )  
MenuIsHiddenOrHasHiddenAncestor( NSInteger menuIndex ) = Boolean  
MenuProperty( NSInteger menuIndex, CFStringRef key ) = CTypeRef  
MenuSetProperty( NSInteger menuIndex, CFStringRef key, CTypeRef value )  
MenuRemoveProperty( NSInteger menuIndex, CFStringRef key )  
MenuRemoveAllProperties( NSInteger menuIndex )
```

### See Also

[MenuItem](#)

### Apple documentation

[NSMenu](#)



---

## MenuItem

### Functions

```
MenuItemAtIndex( NSInteger menuIndex, NSInteger itemIndex ) = MenuItemRef  
MenuItemExists( NSInteger menuIndex, NSInteger itemIndex ) = Boolean
```

#### Enable

```
MenuItemIsEnabled( NSInteger menuIndex, NSInteger itemIndex ) = Boolean  
MenuItemSetEnabled( NSInteger menuIndex, NSInteger itemIndex, Boolean flag )
```

#### Hidden

```
MenuItemIsHidden( NSInteger menuIndex, NSInteger itemIndex ) = Boolean  
MenuItemSetHidden( NSInteger menuIndex, NSInteger itemIndex, Boolean flag )  
MenuItemIsHiddenOrHasHiddenAncestor( NSInteger menuIndex, NSInteger itemIndex ) = Boolean
```

#### Target and action

```
MenuItemSetAction( NSInteger menuIndex, NSInteger itemIndex, CFStringRef actionName )  
MenuItemSetActionCallback( NSInteger menuIndex, NSInteger itemIndex, ptr callback, ptr userData )
```

#### Title

```
MenuItemTitle( NSInteger menuIndex, NSInteger menuItemIndex ) = CFStringRef  
MenuItemSetTitle( NSInteger menuIndex, NSInteger itemIndex, CFStringRef title )  
MenuItemAttributedTitle( NSInteger menuIndex, NSInteger itemIndex ) = CFAttributedStringRef  
MenuItemSetAttributedTitle( NSInteger menuIndex, NSInteger itemIndex, CFAttributedStringRef title )
```

#### Tag

```
MenuItemTag( NSInteger menuIndex, NSInteger itemIndex ) = NSInteger  
MenuItemSetTag( NSInteger menuIndex, NSInteger itemIndex, NSInteger tag )
```

#### State

```
MenuItemState( NSInteger menuIndex, NSInteger itemIndex ) = NSCellStateValue  
MenuItemSetState( NSInteger menuIndex, NSInteger itemIndex, NSControlStateValue state )
```

#### Image

```
MenuItemImage( NSInteger menuIndex, NSInteger itemIndex ) = ImageRef  
MenuItemSetImage( NSInteger menuIndex, NSInteger itemIndex, ImageRef image )  
MenuItemOnStateImage( NSInteger menuIndex, NSInteger itemIndex ) = ImageRef  
MenuItemSetOnStateImage( NSInteger menuIndex, NSInteger itemIndex, ImageRef image )  
MenuItemOffStateImage( NSInteger menuIndex, NSInteger itemIndex ) = ImageRef  
MenuItemSetOffStateImage( NSInteger menuIndex, NSInteger itemIndex, ImageRef image )  
MenuItemMixedStateImage( NSInteger menuIndex, NSInteger itemIndex ) = ImageRef  
MenuItemSetMixedStateImage( NSInteger menuIndex, NSInteger itemIndex, ImageRef image )
```

#### Submenu

```
MenuItemHasSubmenu( NSInteger menuIndex, NSInteger itemIndex ) = Boolean
```

#### Separator

```
MenuItemIsSeparator( NSInteger menuIndex, NSInteger itemIndex ) = Boolean
```

#### KeyEquivalent

```
MenuItemKeyEquivalent( NSInteger menuIndex, NSInteger itemIndex ) = CFStringRef  
MenuItemSetKeyEquivalent( NSInteger menuIndex, NSInteger itemIndex, CFStringRef string )  
MenuItemKeyEquivalentModifierMask( NSInteger menuIndex, NSInteger itemIndex ) = NSUInteger  
MenuItemSetKeyEquivalentModifierMask( NSInteger menuIndex, NSInteger itemIndex, NSUInteger mask )
```

#### User key equivalents

```
MenuItemUsesUserKeyEquivalents = Boolean  
MenuItemUserKeyEquivalent( NSInteger menuIndex, NSInteger itemIndex ) = CFStringRef
```

#### Alternate

```
MenuItemIsAlternate( NSInteger menuIndex, NSInteger itemIndex ) = Boolean  
MenuItemSetAlternate( NSInteger menuIndex, NSInteger itemIndex, Boolean flag )
```

#### Indentation

```
MenuItemIndentationLevel( NSInteger menuIndex, NSInteger itemIndex ) = NSInteger  
MenuItemSetIndentationLevel( NSInteger menuIndex, NSInteger itemIndex, NSInteger level )
```

#### ToolTip

```
MenuItemToolTip( NSInteger menuIndex, NSInteger itemIndex ) = CFStringRef  
MenuItemSetToolTip( NSInteger menuIndex, NSInteger itemIndex, CFStringRef string )
```

#### Highlighted

```
MenuItemIsHighlighted( NSInteger menuIndex, NSInteger itemIndex ) = Boolean
```

#### Instance properties

```
MenuItemAllowsKeyEquivalentWhenHidden( NSInteger menuIndex, NSInteger itemIndex ) = Boolean// macOS 10.13+
```

```
MenuItemSetAllowsKeyEquivalentWhenHidden( NSInteger menuIndex, NSInteger itemIndex, Boolean flag )// macOS 10.13+
```

#### Custom

```
MenuItemSetOnMenuAction( NSInteger menuIndex, NSInteger itemIndex )
```

```
MenuItemProperty( NSInteger menuIndex, NSInteger itemIndex, CFStringRef key ) = CFTypeRef
```

```
MenuItemSetProperty( NSInteger menuIndex, NSInteger itemIndex, CFStringRef key, CFTypeRef value )
```

```
MenuItemRemoveProperty( NSInteger menuIndex, NSInteger itemIndex, CFStringRef key )
```

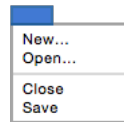
```
MenuItemRemoveAllProperties( NSInteger menuIndex, NSInteger itemIndex )
```

### See Also

[Cocoa Menu](#)

### Apple documentation

[NSMenuItem](#)



## NibMenu

statement

### Syntax

**nibmenu** *nibName*

### Description

The **nibmenu** statement loads a menu from a nib file. Once the menu has loaded, its characteristics can be changed with the **cocoa menu** statement

### Parameters

*nibName*

The name of the nib file (with or without its extension) containing the menu.

### Menu Events

See [cocoa menu](#)

### Functions

See [cocoa menu](#)



## NibPopover

statement

### Syntax

**nibpopover** *tag, nibName*

### Description

The **nibpopover** statement loads a popover from a nib file. Once the popover has loaded, its characteristics can be changed with the **popover** statement

### Parameters

<i>tag</i>	A unique number to identify the popover. The tag value cannot be the same as an existing Carbon or cocoa window.
<i>nibName</i>	The name of the nib file (with or without its extension) containing the popover.

### Functions

See [Popover](#)



---

## NibToolbar

statement

### Syntax

**nibtoolbar** *tag*, *nibName*

### Description

The **nibtoolbar** statement loads a toolbar from a nib file. Once the toolbar has loaded, its characteristics can be changed with the **toolbar** statement

### Parameters

<i>tag</i>	A unique number to identify the toolbar.
<i>nibName</i>	The name of the nib file (with or without its extension) containing the toolbar.

### Functions

See [Toolbar](#)

### See Also

[Toolbar](#), [ToolbarItem](#)



NibView

statement

**Syntax**  
`nibview tag, nibName, identifier, origin, superviewTag`

**Description**  
The **nibview** statement loads a view from a nib file and places in a superview at the specified origin. Once loaded, the view and its subviews can be manipulated with their respective statements.

<b>Parameters</b>	
<i>tag</i>	A number to identify the view. A negative tag hides the view.
<i>nibName</i>	The nib file name (with or without its extension) containing the view.
<i>identifier</i>	The identifier given to the view in the nib file. If this parameter is omitted, the first NSView found in the nib will be used.
<i>origin</i>	The origin of the view in window coordinates. This can be specified in either of two ways: (1) ( <i>x</i> , <i>y</i> ) (2) <a href="#">CGPoint</a> value Optional parameter. Default origin is lower-left corner of superview.
<i>superviewTag</i>	Optional parameter.

**Functions**  
[See View](#)





## NibWindow

statement

### Syntax

**nibwindow** *tag, nibName, wndIdentifier*

### Description

Loads a cocoa window from a nib file. Once the window has loaded, its characteristics can be changed by using the cocoa window statement or by cocoa window functions.

### Parameters

<i>tag</i>	Unique number to identify the window. A negative value hides the window
<i>nibName</i>	The name of the nib file, with or without its extension.
<i>wndIdentifier</i>	The identifier value given to the window in the nib. If this parameter is omitted, the first NSWindow or NSPanel found in the nib will be used.

### Dialog Events

See [Cocoa Window](#)

### Functions

See [Cocoa Window](#)



---

## Notification

### Functions

Create

```
NotificationWithName( NSNotificationName name, CTypeRef object, CFDictionaryRef userInfo ) = NotificationRef
```

Info

```
NotificationName( NotificationRef ref ) = NSNotificationName
```

```
NotificationObject( NotificationRef ref ) = CTypeRef
```

```
NotificationUserInfo( NotificationRef ref ) = CFDictionaryRef
```

### Apple documentation

[NSNotification](#)



---

## NotificationCenter

### Functions

Default center

```
NotificationCenterDefault = NotificationCenterRef
```

Add and remove observers

```
NotificationCenterAddObserver( ptr callback, CFStringRef name, CTypeRef obj )
```

```
NotificationCenterRemoveObserver( ptr callback, CFStringRef name )
```

Post notification

```
NotificationCenterPostNotification( NotificationRef note )
```

```
NotificationCenterPostNotificationName( CFStringRef name, CTypeRef obj, CFDictionaryRef userInfo )
```

### Apple documentation

[NSNotificationCenter](#)



---

## NotificationQueue

### Functions

Create

```
NotificationQueueWithNotificationCenter( NotificationCenterRef ref ) = NotificationQueueRef
```

Default queue

```
NotificationQueueDefault = NotificationQueueRef
```

Managing notifications

```
NotificationQueueEnqueueNotificationWithCoalescing( NotificationQueueRef queue, NotificationRef note, NSPostingStyle style, NSNotificationCoalescing coalesceMask, CFArrayRef modes )
```

```
NotificationQueueEnqueueNotification( NotificationQueueRef queue, NotificationRef note, NSPostingStyle style )
```

```
NotificationQueueDequeueNotification( NotificationQueueRef queue, NotificationRef note, NSNotificationCoalescing coalesceMask )
```

### Apple documentation

[NSNotificationQueue](#)



---

## NSDecimal

### Functions

Create from another decimal

```
NSDecimalCopy( NSDecimal *dcmDest, NSDecimal *dcmSrc )
```

String

```
NSDecimalString( NSDecimal *dcm, CFLocaleRef locale ) = CFStringRef
```

Arithmetic

```
NSDecimalCompact( NSDecimal *dcm )
```

```
NSDecimalAdd( NSDecimal *result, NSDecimal *leftOperand, NSDecimal *rightOperand, NSRoundingMode roundingMode ) =
```

```
NSCalculationError
```

```
NSDecimalSubtract( NSDecimal *result, NSDecimal *leftOperand, NSDecimal *rightOperand, NSRoundingMode roundingMode ) =
```

```
NSCalculationError
```

```
NSDecimalDivide( NSDecimal *result, NSDecimal *leftOperand, NSDecimal *rightOperand, NSRoundingMode roundingMode ) =
```

```
NSCalculationError
```

```
NSDecimalMultiply( NSDecimal *result, NSDecimal *leftOperand, NSDecimal *rightOperand, NSRoundingMode roundingMode ) =
```

```
NSCalculationError
```

```
NSDecimalMultiplyByPowerOf10( NSDecimal *result, NSDecimal *number, short power, NSRoundingMode roundingMode ) =
```

```
NSCalculationError
```

```
NSDecimalRound( NSDecimal *result, NSDecimal *number, NSInteger scale, NSRoundingMode roundingMode ) =
```

```
NSCalculationError
```

```
NSDecimalPower( NSDecimal *result, NSDecimal *number, NSUInteger power, NSRoundingMode roundingMode ) =
```

```
NSCalculationError
```

```
NSDecimalNormalize( NSDecimal *number1, NSDecimal *number2, NSRoundingMode roundingMode ) = NSCalculationError
```

Compare

```
NSDecimalCompare( NSDecimal *dcm1, NSDecimal *dcm1 ) = NSComparisonResult
```

Validation

```
NSDecimalIsNotANumber( NSDecimal *dcm ) = Boolean
```

Convenience

```
NSDecimalWithString( CFStringRef string ) = NSDecimal
```

### Apple documentation

[NSDecimal](#)



---

### NSHFSFileTypes

#### Functions

```
NSFileTypeForHFSTypeCode( OSType hfsFileTypeCode ) = CFStringRef  
NSHFSTypeCodeFromFileType( CFStringRef fileTypeString ) = OSType  
nsHFSTypeOfFileAtURL( CFURLRef url ) = CFStringRef
```



---

## Number

### Functions

Create

```
NumberWithBool( Boolean value ) = CFNumberRef
NumberWithChar( char value ) = CFNumberRef
NumberWithShort( short value ) = CFNumberRef
NumberWithLong( long value ) = CFNumberRef
NumberWithLongLong( SInt64 value ) = CFNumberRef
NumberWithUnsignedChar( unsigned char value ) = CFNumberRef
NumberWithUnsignedShort( unsigned short value ) = CFNumberRef
NumberWithUnsignedLong( unsigned long value ) = CFNumberRef
NumberWithUnsignedLongLong( UInt64 value ) = CFNumberRef
NumberWithFloat( float value ) = CFNumberRef
NumberWithDouble( double value ) = CFNumberRef
```

Numeric values

```
NumberBoolValue( CFNumberRef number ) = Boolean
NumberCharValue( CFNumberRef number ) = char
NumberShortValue( CFNumberRef number ) = short
NumberLongValue( CFNumberRef number ) = long
NumberLongLongValue( CFNumberRef number ) = SInt64
NumberUnsignedCharValue( CFNumberRef number ) = unsigned char
NumberUnsignedShortValue( CFNumberRef number ) = unsigned short
NumberUnsignedLongValue( CFNumberRef number ) = unsigned long
NumberUnsignedLongLongValue( CFNumberRef number ) = UInt64
NumberFloatValue( CFNumberRef number ) = float
NumberDoubleValue( CFNumberRef number ) = double
```

String representation

```
NumberDescriptionWithLocale( CFNumberRef number, CFLocaleRef locale ) = CFStringRef
NumberStringValue( CFNumberRef number ) = CFStringRef
```

Compare

```
NumberCompare( CFNumberRef number1, CFNumberRef number2 ) = NSComparisonResult
NumberIsEqualToNumber( CFNumberRef number1, CFNumberRef number2 ) = Boolean
```

Type information

```
NumberObjCType( CFNumberRef number ) = CFStringRef
```

### Apple documentation

[NSNumber](#)



## NumberFormatter

### Functions

#### Init

```
NumberFormatterInit = NumberFormatterRef// autoreleased
```

#### Configure

```
NumberFormatterBehavior( NumberFormatterRef formatter ) = NSNumberFormatterBehavior
NumberFormatterSetBehavior( NumberFormatterRef formatter, NSNumberFormatterBehavior behavior )
NumberFormatterDefaultBehavior = NSNumberFormatterBehavior
NumberFormatterSetDefaultBehavior( NSNumberFormatterBehavior behavior )
NumberFormatterNumberStyle( NumberFormatterRef formatter ) = NSNumberFormatterStyle
NumberFormatterSetNumberStyle( NumberFormatterRef formatter, NSNumberFormatterStyle style )
NumberFormatterGeneratesDecimalNumbers( NumberFormatterRef formatter ) = Boolean
NumberFormatterSetGeneratesDecimalNumbers( NumberFormatterRef formatter, Boolean flag )
```

#### Convert between numbers and strings

```
NumberFormatterGetObjectValue( NumberFormatterRef formatter, CTypeRef *obj, CFStringRef string, CFRange *range,
ErrorRef *err ) = Boolean
NumberFormatterNumberFromString( NumberFormatterRef formatter, CFStringRef string ) = CFNumberRef
NumberFormatterStringFromNumber( NumberFormatterRef formatter, CFNumberRef number ) = CFStringRef
NumberFormatterLocalizedStringFromNumber( CFNumberRef number, NSNumberFormatterStyle style ) = CFStringRef
```

#### Localization

```
NumberFormatterLocalizesFormat( NumberFormatterRef formatter ) = Boolean
NumberFormatterSetLocalizesFormat( NumberFormatterRef formatter, Boolean flag )
NumberFormatterLocale( NumberFormatterRef formatter ) = CFLocaleRef
NumberFormatterSetLocale( NumberFormatterRef formatter, CFLocaleRef locale )
```

#### Rounding

```
NumberFormatterRoundingIncrement( NumberFormatterRef formatter ) = CFNumberRef
NumberFormatterSetRoundingIncrement( NumberFormatterRef formatter, CFNumberRef number )
NumberFormatterRoundingMode( NumberFormatterRef formatter ) = NSNumberFormatterRoundingMode
NumberFormatterSetRoundingMode( NumberFormatterRef formatter, NSNumberFormatterRoundingMode mode )
```

#### Formats

```
NumberFormatterFormat( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetFormat( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterFormattingContext( NumberFormatterRef formatter ) = NSFormattingContext// macOS 10.10+
NumberFormatterSetFormattingContext( NumberFormatterRef formatter, NSFormattingContext context )// macOS 10.10+
NumberFormatterFormatWidth( NumberFormatterRef formatter ) = NSUInteger
NumberFormatterSetFormatWidth( NumberFormatterRef formatter, NSUInteger value )
NumberFormatterNegativeFormat( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetNegativeWidth( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterPositiveFormat( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetPositiveFormat( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterMultiplier( NumberFormatterRef formatter ) = CFNumberRef
NumberFormatterSetMultiplier( NumberFormatterRef formatter, CFNumberRef number )
```

#### Symbols

```
NumberFormatterPercentSymbol( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetPercentSymbol( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterPerMillSymbol( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetPerMillSymbol( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterMinusSign( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetMinusSign( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterPlusSign( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetPlusSign( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterExponentSymbol( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetExponentSymbol( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterZeroSymbol( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetZeroSymbol( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterNilSymbol( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetNilSymbol( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterNotANumberSymbol( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetNotANumberSymbol( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterNegativeInfinitySymbol( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetNegativeInfinitySymbol( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterPositiveInfinitySymbol( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetPositiveInfinitySymbol( NumberFormatterRef formatter, CFStringRef string )
```



## Currency

```
NumberFormatterCurrencySymbol( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetCurrencySymbol( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterCurrencyCode( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetCurrencyCode( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterInternationalCurrencySymbol( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetInternationalCurrencySymbol( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterCurrencyGroupingSeparator( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetCurrencyGroupingSeparator( NumberFormatterRef formatter, CFStringRef string )
```

## Prefix and suffix

```
NumberFormatterPositivePrefix( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetPositivePrefix( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterPositiveSuffix( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetPositiveSuffix( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterNegativePrefix( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetNegativePrefix( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterNegativeSuffix( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetNegativeSuffix( NumberFormatterRef formatter, CFStringRef string )
```

## Display

```
NumberFormatterTextAttributesForNegativeValues( NumberFormatterRef formatter ) = CFDictionaryRef
NumberFormatterSetTextAttributesForNegativeValues( NumberFormatterRef formatter, CFDictionaryRef attributes )
NumberFormatterTextAttributesForPositiveValues( NumberFormatterRef formatter ) = CFDictionaryRef
NumberFormatterSetTextAttributesForPositiveValues( NumberFormatterRef formatter, CFDictionaryRef attributes )
NumberFormatterAttributedStringForZero( NumberFormatterRef formatter ) = CFAttributedStringRef
NumberFormatterSetAttributedStringForZero( NumberFormatterRef formatter, CFAttributedStringRef aString )
NumberFormatterTextAttributesForZero( NumberFormatterRef formatter ) = CFDictionaryRef
NumberFormatterSetTextAttributesForZero( NumberFormatterRef formatter, CFDictionaryRef attributes )
NumberFormatterAttributedStringForNil( NumberFormatterRef formatter ) = CFAttributedStringRef
NumberFormatterSetAttributedStringForNil( NumberFormatterRef formatter, CFAttributedStringRef aString )
NumberFormatterTextAttributesForNil( NumberFormatterRef formatter ) = CFDictionaryRef
NumberFormatterSetTextAttributesForNil( NumberFormatterRef formatter, CFDictionaryRef attributes )
NumberFormatterAttributedStringForNotANumber( NumberFormatterRef formatter ) = CFAttributedStringRef
NumberFormatterSetAttributedStringForNotANumber( NumberFormatterRef formatter, CFAttributedStringRef aString )
NumberFormatterTextAttributesForNotANumber( NumberFormatterRef formatter ) = CFDictionaryRef
NumberFormatterSetTextAttributesForNotANumber( NumberFormatterRef formatter, CFDictionaryRef attributes )
NumberFormatterTextAttributesForPositiveInfinity( NumberFormatterRef formatter ) = CFDictionaryRef
NumberFormatterSetTextAttributesForPositiveInfinity( NumberFormatterRef formatter, CFDictionaryRef attributes )
NumberFormatterTextAttributesForNegativeInfinity( NumberFormatterRef formatter ) = CFDictionaryRef
NumberFormatterSetTextAttributesForNegativeInfinity( NumberFormatterRef formatter, CFDictionaryRef attributes )
```

## Separators and grouping

```
NumberFormatterGroupingSeparator( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetGroupingSeparator( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterUsesGroupingSeparator( NumberFormatterRef formatter ) = Boolean
NumberFormatterSetUsesGroupingSeparator( NumberFormatterRef formatter, Boolean flag )
NumberFormatterThousandSeparator( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetThousandSeparator( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterHasThousandSeparators( NumberFormatterRef formatter ) = Boolean
NumberFormatterSetHasThousandSeparators( NumberFormatterRef formatter, Boolean flag )
NumberFormatterDecimalSeparator( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetDecimalSeparator( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterAlwaysShowsDecimalSeparator( NumberFormatterRef formatter ) = Boolean
NumberFormatterSetAlwaysShowsDecimalSeparator( NumberFormatterRef formatter, Boolean flag )
NumberFormatterCurrencyDecimalSeparator( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetCurrencyDecimalSeparator( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterGroupingSize( NumberFormatterRef formatter ) = NSUInteger
NumberFormatterSetGroupingSize( NumberFormatterRef formatter, NSUInteger value )
NumberFormatterSecondaryGroupingSize( NumberFormatterRef formatter ) = NSUInteger
NumberFormatterSetSecondaryGroupingSize( NumberFormatterRef formatter, NSUInteger value )
```

## Padding

```
NumberFormatterPaddingCharacter( NumberFormatterRef formatter ) = CFStringRef
NumberFormatterSetPaddingCharacter( NumberFormatterRef formatter, CFStringRef string )
NumberFormatterPaddingPosition( NumberFormatterRef formatter ) = NSNumberFormatterPadPosition
NumberFormatterSetPaddingPosition( NumberFormatterRef formatter, NSNumberFormatterPadPosition position )
```

## Input and output attributes

```
NumberFormatterAllowsFloats( NumberFormatterRef formatter ) = Boolean
NumberFormatterSetAllowsFloats( NumberFormatterRef formatter, Boolean flag )
NumberFormatterMinimum( NumberFormatterRef formatter ) = CFNumberRef
NumberFormatterSetMinimum( NumberFormatterRef formatter, CFNumberRef number )
NumberFormatterMaximum( NumberFormatterRef formatter ) = CFNumberRef
NumberFormatterSetMaximum( NumberFormatterRef formatter, CFNumberRef number )
```

## Integer and fraction digits

```
NumberFormatterMinimumIntegerDigits( NumberFormatterRef formatter ) = NSUInteger
NumberFormatterSetMinimumIntegerDigits( NumberFormatterRef formatter, NSUInteger value )
NumberFormatterMaximumIntegerDigits( NumberFormatterRef formatter ) = NSUInteger
NumberFormatterSetMaximumIntegerDigits( NumberFormatterRef formatter, NSUInteger value )
```

```
NumberFormatterMinimumFractionDigits( NumberFormatterRef formatter ) = NSUInteger
NumberFormatterSetMinimumFractionDigits( NumberFormatterRef formatter, NSUInteger value )
NumberFormatterMaximumFractionDigits( NumberFormatterRef formatter ) = NSUInteger
NumberFormatterSetMaximumFractionDigits( NumberFormatterRef formatter, NSUInteger value )
```

#### Significant digits

```
NumberFormatterUsesSignificantDigits( NumberFormatterRef formatter ) = Boolean
NumberFormatterSetUsesSignificantDigits( NumberFormatterRef formatter, Boolean flag )
NumberFormatterMinimumSignificantDigits( NumberFormatterRef formatter ) = NSUInteger
NumberFormatterSetMinimumSignificantDigits( NumberFormatterRef formatter, NSUInteger value )
NumberFormatterMaximumSignificantDigits( NumberFormatterRef formatter ) = NSUInteger
NumberFormatterSetMaximumSignificantDigits( NumberFormatterRef formatter, NSUInteger value )
```

#### Leniency behavior

```
NumberFormatterIsLenient( NumberFormatterRef formatter ) = Boolean
NumberFormatterSetLenient( NumberFormatterRef formatter, Boolean flag )
```

#### Validation of partial numeric

```
NumberFormatterIsPartialStringValidationEnabled( NumberFormatterRef formatter ) = Boolean
NumberFormatterSetPartialStringValidationEnabled( NumberFormatterRef formatter, Boolean flag )
```

## Apple documentation

[NSNumberFormatter](#)



---

## Object

### Functions

Create, copy, dealloc

```
ObjectCopy( CTypeRef obj ) = ptr// autoreleased  
ObjectMutableCopy( CTypeRef obj ) = ptr// autoreleased
```

Identify class

```
ObjectIsSubclassOfClass( CTypeRef obj, ClassRef class ) = Boolean
```

Sending messages

```
ObjectPerformSelectorOnMainThread( CTypeRef obj, CFStringRef selector, CTypeRef withObj, Boolean waitUntilDone )  
ObjectPerformSelectorInBackground( CTypeRef obj, CFStringRef selector, CTypeRef withObj )
```

Instance methods

```
ObjectIsEqual( CTypeRef obj1, CTypeRef obj2 ) = Boolean  
ObjectSetValueForKey( CTypeRef obj, CTypeRef value, CFStringRef key )  
ObjectValueForKey( CTypeRef obj, CFStringRef key ) = CTypeRef
```

Protocol methods

```
ObjectClass( CTypeRef obj ) = ClassRef  
ObjectSuperclass( CTypeRef obj ) = ClassRef  
ObjectIsKindOfClass( CTypeRef obj, ClassRef class ) = Boolean
```

Custom

```
ObjectCallFunctionOnMainThread( ptr fnAddress, CTypeRef withObj, Boolean waitUntilDone )  
ObjectCallFunctionInBackground( ptr fnAddress, CTypeRef withObj )  
ObjectProperty( CTypeRef obj, CFStringRef key ) = CTypeRef  
ObjectSetProperty( CTypeRef obj, CFStringRef key, CTypeRef value )  
ObjectRemoveProperty( CTypeRef obj, CFStringRef key )  
ObjectRemoveAllProperties( CTypeRef obj )
```

### Apple documentation

[NSObject](#)



---

### ObjCRuntime

#### Functions

Type lookup

```
ClassFromString( CFStringRef string ) = ClassRef  
StringFromClass( ClassRef class ) = CFStringRef  
SelectorFromString( CFStringRef string ) = SELRef  
StringFromSelector( SELRef selector ) = CFStringRef
```

#### Apple documentation

[NSObjCRuntime](#)



## OpenPanel

function/statement

### Syntax

```
[1]  
url | array = openpanel tag, msg, fileTypes, prompt, dirURL, sheetFlag
```

```
[2]  
openpanel tag, msg, fileTypes, prompt, dirURL, sheetFlag
```

### Description

Use this statement to:

- Build and run a new openpanel;
- Run an existing openpanel;
- Alter an existing openpanel's characteristics.

### Parameters

<i>tag</i>	Unique number to identify the openpanel. A negative value builds the panel invisibly and doesn't run it until the statement is executed with a positive value. Note: Once the panel has been dismissed, its tag value is no longer valid.
<i>msg</i>	The message text.
<i>fileTypes</i>	The allowed file types. A semicolon-delimited string or array of types.
<i>prompt</i>	The prompt of the default button.
<i>dirURL</i>	The directory shown in the panel.
<i>sheetFlag</i>	If this param is <code>_true</code> , the panel will be a sheet attached to the current output window. Note: A sheet does not return a response value from the openpanel statement, and instead posts a <code>_openPanelDidEnd</code> event to the user on dialog function.

### Return value (non-sheet only)

If the panel is configured to allow multiple selection, the return value is an array of URLs, otherwise it is a URL.

### Dialog Events

Event	Description
<code>_openPanelDidEnd</code>	This event is triggered when an openpanel sheet is dismissed.

### Functions

```
OpenPanelWithTag( NSInteger tag ) = OpenPanelRef  
OpenPanelExists( NSInteger tag ) = Boolean
```

```
Init  
OpenPanelInit( NSInteger tag ) = OpenPanelRef// autoreleased
```

```
Configure  
OpenPanelCanChooseFiles( NSInteger tag ) = Boolean  
OpenPanelSetCanChooseFiles( NSInteger tag, Boolean flag )  
OpenPanelCanChooseDirectories( NSInteger tag ) = Boolean  
OpenPanelSetCanChooseDirectories( NSInteger tag, Boolean flag )  
OpenPanelResolvesAliases( NSInteger tag ) = Boolean  
OpenPanelSetResolvesAliases( NSInteger tag, Boolean flag )  
OpenPanelAllowsMultipleSelection( NSInteger tag ) = Boolean  
OpenPanelSetAllowsMultipleSelection( NSInteger tag, Boolean flag )
```

```
User selection  
OpenPanelURLs( NSInteger tag ) = CFArrayRef
```

```
Ubiquitous documents  
OpenPanelCanDownloadUbiquitousContents( NSInteger tag ) = Boolean// macOS 10.10+  
OpenPanelSetCanDownloadUbiquitousContents( NSInteger tag, Boolean flag )// macOS 10.10+  
OpenPanelCanResolveUbiquitousConflicts( NSInteger tag ) = Boolean// macOS 10.10+  
OpenPanelSetCanResolveUbiquitousConflicts( NSInteger tag, Boolean flag )// macOS 10.10+
```

#### Instance properties

```
OpenPanelIsAccessoryViewDisclosed( NSInteger tag ) = Boolean// macOS 10.11+
OpenPanelSetAccessoryViewDisclosed( NSInteger tag, Boolean flag )// macOS 10.11+
```

#### Convenience (duplicates of SavePanel functions)

##### Configure

```
OpenPanelAccessoryView( NSInteger tag ) = ViewRef
OpenPanelSetAccessoryView( NSInteger tag, ViewRef ref )
OpenPanelTitle( NSInteger tag ) = CFStringRef
OpenPanelSetTitle( NSInteger tag, CFStringRef title )
OpenPanelPrompt( NSInteger tag ) = CFStringRef
OpenPanelSetPrompt( NSInteger tag, CFStringRef prompt )
OpenPanelMessage( NSInteger tag ) = CFStringRef
OpenPanelSetMessage( NSInteger tag, CFStringRef message )
OpenPanelCanCreateDirectories( NSInteger tag ) = Boolean
OpenPanelSetCanCreateDirectories( NSInteger tag, Boolean flag )
OpenPanelShowsHiddenFiles( NSInteger tag ) = Boolean
OpenPanelSetShowsHiddenFiles( NSInteger tag, Boolean flag )
```

##### Panel content

```
OpenPanelDirectoryURL( NSInteger tag ) = CFURLRef
OpenPanelSetDirectoryURL( NSInteger tag, CFURLRef url )
OpenPanelAllowedFileTypes( NSInteger tag ) = CFArrayRef
OpenPanelSetAllowedFileTypes( NSInteger tag, CFArrayRef fileTypes )
OpenPanelAllowsOtherFileTypes( NSInteger tag ) = Boolean
OpenPanelSetAllowsOtherFileTypes( NSInteger tag, Boolean flag )
OpenPanelTreatsFilePackagesAsDirectories( NSInteger tag ) = Boolean
OpenPanelSetTreatsFilePackagesAsDirectories( NSInteger tag, Boolean flag )
```

##### Running

```
OpenPanelValidateVisibleColumns( NSInteger tag )
```

##### User selection

```
OpenPanelURL( NSInteger tag ) = CFURLRef
```

##### Actions

```
OpenPanelOK( NSInteger tag )
OpenPanelCancel( NSInteger tag )
```

##### Custom

```
OpenPanelRemove( NSInteger tag )
```

## Apple documentation

[NSOpenPanel](#)



---

## Operation

### Functions

Init

```
OperationInit = OperationRef// autoreleased
```

Executing

```
OperationStart( OperationRef op )
```

```
OperationMain( OperationRef op )
```

```
OperationSetCompletionFunction( OperationRef op, ptr fnAddress )// wrapper for setCompletionBlock
```

Cancelling

```
OperationCancel( OperationRef op )
```

Status

```
OperationIsCancelled( OperationRef op ) = Boolean
```

```
OperationIsExecuting( OperationRef op ) = Boolean
```

```
OperationIsFinished( OperationRef op ) = Boolean
```

```
OperationIsConcurrent( OperationRef op ) = Boolean
```

```
OperationIsAsynchronous( OperationRef op ) = Boolean
```

```
OperationIsReady( OperationRef op ) = Boolean
```

```
OperationName( OperationRef op ) = CFStringRef
```

```
OperationSetName( OperationRef op, CFStringRef name )
```

Dependencies

```
OperationAddDependency( OperationRef op1, OperationRef op2 )
```

```
OperationRemoveDependency( OperationRef op1, OperationRef op2 )
```

```
OperationDependencies( OperationRef op ) = CFArrayRef
```

Execution priority

```
OperationQualityOfService( OperationRef op ) = NSQualityOfService
```

```
OperationSetQualityOfService( OperationRef op, NSQualityOfService qos )
```

```
OperationQueuePriority( OperationRef op ) = NSOperationQueuePriority
```

```
OperationSetQueuePriority( OperationRef op, NSOperationQueuePriority priority )
```

Waiting

```
OperationWaitUntilFinished( OperationRef op )
```

### Apple documentation

[NSOperation](#)



---

## OperationQueue

### Functions

Init

```
OperationQueueInit = OperationQueueRef// autoreleased
```

Get specific queues

```
OperationQueueMain = OperationQueueRef
OperationQueueCurrent = OperationQueueRef
```

Managing operations

```
OperationQueueAddOperation( OperationQueueRef queue, OperationRef op )
OperationQueueAddOperations( OperationQueueRef queue, CFArrayRef ops, Boolean waitUntilFinished )
OperationQueueAddOperationWithFunction( OperationQueueRef queue, ptr fnAddress )// wrapper for
addOperationWithBlock:
OperationQueueOperations( OperationQueueRef queue ) = CFArrayRef
OperationQueueOperationCount( OperationQueueRef queue ) = NSUInteger
OperationQueueCancelAllOperations( OperationQueueRef queue )
OperationQueueWaitUntilAllOperationsAreFinished( OperationQueueRef queue )
```

Managing execution

```
OperationQueueQualityOfService( OperationQueueRef queue ) = NSQualityOfService
OperationQueueSetQualityOfService( OperationQueueRef queue, NSQualityOfService qos )
OperationQueueMaxConcurrentOperationCount( OperationQueueRef queue ) = NSInteger
OperationQueueSetMaxConcurrentOperationCount( OperationQueueRef queue, NSInteger count )
```

Suspending

```
OperationQueueIsSuspended( OperationQueueRef queue ) = Boolean
OperationQueueSetSuspended( OperationQueueRef queue, Boolean flag )
```

Configure

```
OperationQueueName( OperationQueueRef queue ) = CFStringRef
OperationQueueSetName( OperationQueueRef queue, CFStringRef name )
```

### Apple documentation

[NSOperationQueue](#)





## OrderedSet

### Functions

#### Create

```
OrderedSetWithArray( CFArrayRef array ) = OrderedSetRef
OrderedSetWithArrayRange( CFArrayRef array, CFRange range, Boolean copyItems ) = OrderedSetRef
OrderedSetWithObject( CTypeRef obj ) = OrderedSetRef
OrderedSetWithObjects( CTypeRef obj, ... ) = OrderedSetRef // a comma-separated list of objects, ending with
NULL
OrderedSetWithSet( CFSetRef set ) = OrderedSetRef
```

#### Count

```
OrderedSetCount( OrderedSetRef os ) = NSUInteger
```

#### Accessing set members

```
OrderedSetContainsObject( OrderedSetRef os, CTypeRef obj ) = Boolean
OrderedSetFirstObject( OrderedSetRef os ) = CTypeRef
OrderedSetLastObject( OrderedSetRef os ) = CTypeRef
OrderedSetObjectAtIndex( OrderedSetRef os, NSUInteger index ) = CTypeRef
OrderedSetObjectsAtIndexes( OrderedSetRef os, IndexSetRef indexes ) = CFArrayRef
OrderedSetIndexOfObject( OrderedSetRef os, CTypeRef obj ) = NSUInteger
```

#### Observing

```
OrderedSetIsEqualToOrderedSet( OrderedSetRef os1, OrderedSetRef os2 ) = Boolean
OrderedSetIntersectsOrderedSet( OrderedSetRef os1, OrderedSetRef os2 ) = Boolean
OrderedSetIntersectsSet( OrderedSetRef os, CFSetRef set ) = Boolean
OrderedSetIsSubsetOfOrderedSet( OrderedSetRef os1, OrderedSetRef os2 ) = Boolean
OrderedSetIsSubsetOfSet( OrderedSetRef os, CFSetRef set ) = Boolean
```

#### Sorted array

```
OrderedSetSortedArrayUsingDescriptors( OrderedSetRef os, CFArrayRef descriptors ) = CFArrayRef
```

#### Description

```
OrderedSetDescription( OrderedSetRef os ) = CFStringRef
```

#### Converting

```
OrderedSetArray( OrderedSetRef os ) = CFArrayRef
OrderedSetSet( OrderedSetRef os ) = CFSetRef
```

#### • Mutable ordered set •

##### Create

```
OrderedSetWithCapacity( NSUInteger size ) = MutableOrderedSetRef
```

##### Add, remove, reorder

```
OrderedSetAddObject( MutableOrderedSetRef os, CTypeRef obj )
OrderedSetAddObjectsFromArray( MutableOrderedSetRef os, CFArrayRef array )
OrderedSetInsertObjectAtIndex( MutableOrderedSetRef os, CTypeRef obj, NSUInteger index )
OrderedSetInsertObjectsAtIndexes( MutableOrderedSetRef os, CFArrayRef objects, IndexSetRef indexes )
OrderedSetRemoveObject( MutableOrderedSetRef os, CTypeRef obj )
OrderedSetRemoveObjectAtIndex( MutableOrderedSetRef os, NSUInteger index )
OrderedSetRemoveObjectsAtIndexes( MutableOrderedSetRef os, IndexSetRef indexes )
OrderedSetRemoveObjectsFromArray( MutableOrderedSetRef os, CFArrayRef array )
OrderedSetRemoveObjectsInRange( MutableOrderedSetRef os, CFRange range )
OrderedSetRemoveAllObjects( MutableOrderedSetRef os )
OrderedSetReplaceObjectAtIndex( MutableOrderedSetRef os, CTypeRef obj, NSUInteger index )
OrderedSetReplaceObjectsAtIndexes( MutableOrderedSetRef os, CFArrayRef objects, IndexSetRef indexes )
OrderedSetSetObjectAtIndex( MutableOrderedSetRef os, CTypeRef obj, NSUInteger index )
OrderedSetMoveObjectsAtIndexesToIndex( MutableOrderedSetRef os, IndexSetRef indexes, NSUInteger index )
OrderedSetExchangeObjectAtIndex( MutableOrderedSetRef os, NSUInteger index1, NSUInteger index2 )
```

##### Sorting

```
OrderedSetSortUsingDescriptors( MutableOrderedSetRef os, CFArrayRef descriptors )
```

##### Combining and recombining

```
OrderedSetIntersectOrderedSet( MutableOrderedSetRef os1, OrderedSetRef os2 )
OrderedSetIntersectSet( MutableOrderedSetRef os, CFSetRef set )
OrderedSetMinusOrderedSet( MutableOrderedSetRef os1, OrderedSetRef os2 )
OrderedSetMinusSet( MutableOrderedSetRef os, CFSetRef set )
OrderedSetUnionOrderedSet( MutableOrderedSetRef os1, OrderedSetRef os2 )
OrderedSetUnionSet( MutableOrderedSetRef os, CFSetRef set )
```

## Apple documentation

[NSOrderedSet](#)

[NSMutableOrderedSet](#)



---

## Orthography

### Functions

Create

```
OrthographyDefaultForLanguage( CFStringRef language ) = OrthographyRef// macOS 10.13+
```

```
OrthographyWithDominantScript( CFStringRef script, CFDictionaryRef map ) = OrthographyRef
```

Correspondences between languages and scripts

```
OrthographyLanguageMap( OrthographyRef ref ) = CFDictionaryRef
```

```
OrthographyDominantLanguage( OrthographyRef ref ) = CFStringRef
```

```
OrthographyDominantScript( OrthographyRef ref ) = CFStringRef
```

```
OrthographyDominantLanguageForScript( OrthographyRef ref, CFStringRef script ) = CFStringRef
```

```
OrthographyLanguagesForScript( OrthographyRef ref, CFStringRef script ) = CFArrayRef
```

```
OrthographyAllScripts( OrthographyRef ref ) = CFArrayRef
```

```
OrthographyAllLanguages( OrthographyRef ref ) = CFArrayRef
```



## OutlineView

### Description

The **outlineview** must be view-based and created in a nib window. Requires the Cocoa runtime ([CocoaInit](#)).

### Dialog Events

[\\_btnClick](#)  
[\\_outlineViewDoubleClick](#)  
[\\_outlineViewSelectionDidChange](#)

### Functions

Note: OutlineView inherits from TableView, so shares many of its functions (see TableView.incl)

```
OutlineViewWithTag( NSInteger tag ) = OutlineViewRef  
OutlineViewExists( NSInteger tag ) = Boolean
```

Init  

```
OutlineViewInit( NSInteger tag, CGRect r ) = OutlineViewRef// autoreleased
```

Data source object  

```
OutlineViewDataSource( NSInteger tag ) = CTypeRef  
OutlineViewSetDataSource( NSInteger tag, CTypeRef ref )
```

Expandability  

```
OutlineViewIsItemExpandable( NSInteger tag, OVItemRef item ) = Boolean  
OutlineViewIsItemExpanded( NSInteger tag, OVItemRef item ) = Boolean
```

Expand and collapse  

```
OutlineViewExpandItem( NSInteger tag, OVItemRef item )  
OutlineViewExpandItemAndChildren( NSInteger tag, OVItemRef item, Boolean expandChildren )  
OutlineViewCollapseItem( NSInteger tag, OVItemRef item )  
OutlineViewCollapseItemAndChildren( NSInteger tag, OVItemRef item, Boolean collapseChildren )
```

Redisplay info  

```
OutlineViewReloadItem( NSInteger tag, OVItemRef item )  
OutlineViewReloadItemAndChildren( NSInteger tag, OVItemRef item, Boolean reloadChildren )
```

Converting between items and rows  

```
OutlineViewItemAtRow( NSInteger tag, NSInteger row ) = OVItemRef  
OutlineViewRowForItem( NSInteger tag, OVItemRef item ) = NSInteger
```

Indentation  

```
OutlineViewLevelForItem( NSInteger tag, OVItemRef item ) = NSInteger  
OutlineViewLevelForRow( NSInteger tag, NSInteger row ) = NSInteger
```

Related items  

```
OutlineViewParentForItem( NSInteger tag, OVItemRef item ) = OVItemRef  
OutlineViewChildIndexForItem( NSInteger tag, OVItemRef item ) = NSInteger  
OutlineViewChildOfItem( NSInteger tag, NSInteger index, OVItemRef item ) = OVItemRef  
OutlineViewNumberOfChildrenOfItem( NSInteger tag, OVItemRef item ) = NSInteger
```

Delegate object  

```
OutlineViewDelegate( NSInteger tag ) = CTypeRef  
OutlineViewSetDelegate( NSInteger tag, CTypeRef ref )
```

Custom  

```
OutlineViewData( NSInteger tag ) = CFMutableArrayRef  
OutlineViewSetData( NSInteger tag, CFMutableArrayRef array )  
OutlineViewReloadData( NSInteger tag )
```

## OutlineItem Functions

### Create

```
OutlineItemWithString( CFStringRef string, ImageRef image, Boolean isGroup ) = OutlineItemRef  
OutlineItemWithColumns( CFMutableDictionaryRef columns, Boolean isGroup ) = OutlineItemRef
```

### Info

```
OutlineItemString( OutlineItemRef item ) = CFStringRef  
OutlineItemImage( OutlineItemRef item ) = ImageRef  
OutlineItemIsGroup( OutlineItemRef item ) = Boolean  
OutlineItemChildren( OutlineItemRef item ) = CFMutableArrayRef  
OutlineItemColumns( OutlineItemRef item ) = CFMutableDictionaryRef
```

### Add

```
OutlineItemAddChildWithString( OutlineItemRef parItem, CFStringRef string, ImageRef image, Boolean isGroup ) =  
OutlineItemRef  
OutlineItemAddChild( OutlineItemRef parItem, OutlineItemRef childItem )  
OutlineItemAddChildColumns( OutlineItemRef parItem, CFMutableDictionaryRef columns, Boolean isGroup )
```

### Insert

```
OutlineItemInsertChildWithString( OutlineItemRef parItem, CFStringRef string, ImageRef image, Boolean isGroup,  
NSUInteger index ) = OutlineItemRef  
OutlineItemInsertChild( OutlineItemRef parItem, OutlineItemRef childItem, NSUInteger index )  
OutlineItemInsertChildColumns( OutlineItemRef parItem, CFDictionaryRef columns, Boolean isGroup, NSUInteger index )
```

### Remove

```
OutlineItemRemoveChildAtIndex( OutlineItemRef parItem, NSUInteger index )  
OutlineItemRemoveChild( OutlineItemRef parItem, OutlineItemRef childItem )
```

## Apple documentation

[NSOutlineView](#)



---

## PageLayout

### Functions

Init

```
PageLayoutInit = PageLayoutRef// autoreleased
```

Dialog

```
PageLayoutBeginSheet( PageLayoutRef pageLayout, PrintInfoRef info, NSInteger wndTag, ptr callback, ptr contextInfo )
```

```
PageLayoutRunModal( PageLayoutRef pageLayout ) = NSInteger
```

```
PageLayoutRunModalWithPrintInfo( PageLayoutRef pageLayout, PrintInfoRef info ) = NSInteger
```

Customize

```
PageLayoutAddAccessoryController( PageLayoutRef pageLayout, ViewControllerRef vc )
```

```
PageLayoutRemoveAccessoryController( PageLayoutRef pageLayout, ViewControllerRef vc )
```

```
PageLayoutAccessoryControllers( PageLayoutRef pageLayout ) = CFArrayRef
```

Printinfo

```
PageLayoutPrintInfo( PageLayoutRef pageLayout ) = PrintInfoRef
```

### Apple documentation

[NSPageLayout](#)



Panel

statement

Syntax

```
panel tag, title, rect, style
```

Description

Use this statement to:

- Create a new panel;
- Activate and bring to the front an existing panel;
- Make an existing panel visible or invisible;
- Alter the title or rectangle of an existing panel.

Parameters

tag	Unique number to identify the panel. A negative value hides the panel.
title	The panel title.
rect	Origin and size of the panel's content area in screen coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) CGRect value
style	The panel's style can be any of the options described below. They can be combined using '+'. NSBorderlessWindowMask NSTitledWindowMask (default) NSClosableWindowMask (default) NSMiniaturizableWindowMask NSResizableWindowMask (default) NSTexturedBackgroundWindowMask NSUnifiedTitleAndToolbarWindowMask NSFullScreenWindowMask NSFullSizeContentViewWindowMask NSUtilityWindowMask (default) NSDocModalWindowMask NSNonactivatingPanelMask NSHUDWindowMask

Dialog Events

See [cocoa window](#)

Functions

See [cocoa window](#)

```
PanelWithTag( NSInteger tag ) = PanelRef

Configure
PanelSetFloatingPanel( NSInteger tag, Boolean flag )
PanelSetBecomesKeyOnlyIfNeeded( NSInteger tag, Boolean flag )
PanelSetWorksWhenModal( NSInteger tag, Boolean flag )
```

Apple documentation

[NSPanel](#)



---

## PanGestureRecognizer

### Functions

Init

```
PanGestureRecognizerInit( ptr callback, ptr userData ) = PanGestureRecognizerRef// autoreleased
```

Configure

```
PanGestureRecognizerButtonMask( PanGestureRecognizerRef ref ) = NSUInteger  
PanGestureRecognizerSetButtonMask( PanGestureRecognizerRef ref, NSUInteger mask )
```

Tracking location and velocity

```
PanGestureRecognizerTranslationInView( PanGestureRecognizerRef ref, NSInteger tag ) = CGPoint  
PanGestureRecognizerSetTranslationInView( PanGestureRecognizerRef ref, CGPoint duration, NSInteger tag )  
PanGestureRecognizerVelocityInView( PanGestureRecognizerRef ref, NSInteger tag ) = CGPoint
```

Instance properties

```
PanGestureRecognizerNumberOfTouchesRequired( PanGestureRecognizerRef ref ) = NSInteger// macOS 10.12.2  
PanGestureRecognizerSetNumberOfTouchesRequired( PanGestureRecognizerRef ref, NSInteger touches )// macOS 10.12.2
```

### Apple documentation

[NSPanGestureRecognizer](#)





## ParagraphStyle

### Functions

Create

```
ParagraphStyleDefaultStyle = ParagraphStyleRef
```

Style

```
ParagraphStyleAlignment( ParagraphStyleRef ref ) = NSTextAlignment
ParagraphStyleFirstLineHeadIndent( ParagraphStyleRef ref ) = CGFloat
ParagraphStyleHeadIndent( ParagraphStyleRef ref ) = CGFloat
ParagraphStyleTailIndent( ParagraphStyleRef ref ) = CGFloat
ParagraphStyleLineHeightMultiple( ParagraphStyleRef ref ) = CGFloat
ParagraphStyleMaximumLineHeight( ParagraphStyleRef ref ) = CGFloat
ParagraphStyleMinimumLineHeight( ParagraphStyleRef ref ) = CGFloat
ParagraphStyleLineSpacing( ParagraphStyleRef ref ) = CGFloat
ParagraphStyleParagraphSpacing( ParagraphStyleRef ref ) = CGFloat
ParagraphStyleParagraphSpacingBefore( ParagraphStyleRef ref ) = CGFloat
```

Tab info

```
ParagraphStyleTabStops( ParagraphStyleRef ref ) = CFArrayRef
ParagraphStyleDefaultTabInterval( ParagraphStyleRef ref ) = CGFloat
```

Text block and list info

```
ParagraphStyleTextBlocks( ParagraphStyleRef ref ) = CFArrayRef
ParagraphStyleTextLists( ParagraphStyleRef ref ) = CFArrayRef
```

Line breaking info

```
ParagraphStyleLineBreakMode( ParagraphStyleRef ref ) = NSLineBreakMode
ParagraphStyleHyphenationFactor( ParagraphStyleRef ref ) = float
ParagraphStyleTighteningFactorForTruncation( ParagraphStyleRef ref ) = float
ParagraphStyleAllowsDefaultTighteningForTruncation( ParagraphStyleRef ref ) = Boolean// macOS 10.11+
```

HTML header level

```
ParagraphStyleHeaderLevel( ParagraphStyleRef ref ) = NSInteger
```

Writing direction

```
ParagraphStyleDefaultWritingDirectionForLanguage( CFStringRef languageName ) = NSWritingDirection
ParagraphStyleBaseWritingDirection( ParagraphStyleRef ref ) = NSWritingDirection
```

• Mutable paragraph style •

Init

```
ParagraphStyleInit = MutableParagraphStyleRef// autoreleased
```

Tyle info

```
ParagraphStyleSet( MutableParagraphStyleRef ref, ParagraphStyleRef style )
ParagraphStyleSetAlignment( MutableParagraphStyleRef ref, NSTextAlignment alignment )
ParagraphStyleSetFirstLineHeadIndent( MutableParagraphStyleRef ref, CGFloat indent )
ParagraphStyleSetHeadIndent( MutableParagraphStyleRef ref, CGFloat indent )
ParagraphStyleSetTailIndent( MutableParagraphStyleRef ref, CGFloat indent )
ParagraphStyleSetLineBreakMode( MutableParagraphStyleRef ref, NSLineBreakMode lineBreakMode )
ParagraphStyleSetMaximumLineHeight( MutableParagraphStyleRef ref, CGFloat height )
ParagraphStyleSetMinimumLineHeight( MutableParagraphStyleRef ref, CGFloat height )
ParagraphStyleSetLineSpacing( MutableParagraphStyleRef ref, CGFloat spacing )
ParagraphStyleSetParagraphSpacing( MutableParagraphStyleRef ref, CGFloat spacing )
ParagraphStyleSetParagraphSpacingBefore( MutableParagraphStyleRef ref, CGFloat spacing )
ParagraphStyleSetBaseWritingDirection( MutableParagraphStyleRef ref, NSWritingDirection direction )
ParagraphStyleSetLineHeightMultiple( MutableParagraphStyleRef ref, CGFloat height )
```

Tab info

```
ParagraphStyleAddTabStop( MutableParagraphStyleRef ref, TextTabRef textTab )
ParagraphStyleRemoveTabStop( MutableParagraphStyleRef ref, TextTabRef textTab )
ParagraphStyleSetTabStops( MutableParagraphStyleRef ref, CFArrayRef tabStops )
ParagraphStyleSetDefaultTabInterval( MutableParagraphStyleRef ref, CGFloat interval )
```

Text blocks and lists

```
ParagraphStyleSetTextBlocks( MutableParagraphStyleRef ref, CFArrayRef blocks )
ParagraphStyleSetTextLists( MutableParagraphStyleRef ref, CFArrayRef lists )
```

Hyphenation factor

```
ParagraphStyleSetHyphenationFactor( MutableParagraphStyleRef ref, float factor )
ParagraphStyleSetTighteningFactorForTruncation( ParagraphStyleRef ref, float factor )
ParagraphStyleSetAllowsDefaultTighteningForTruncation( ParagraphStyleRef ref, Boolean flag )// macOS 10.11+
```

HTML header level

```
ParagraphStyleSetHeaderLevel( ParagraphStyleRef ref, NSInteger level )
```

## **Apple documentation**

[NSParagraphStyle](#)

[NSMutableParagraphStyle](#)



## Pasteboard

---

### Functions

Create

```
PasteboardGeneral = CocoaPasteboardRef
```

```
PasteboardWithName( CFStringRef name ) = CocoaPasteboardRef
```

```
PasteboardWithUniqueName = CocoaPasteboardRef
```

```
PasteboardReleaseGlobally( CocoaPasteboardRef pb )
```

 Call on temporary, privately named pasteboards only. Never call this on a standard pasteboard

Writing data

```
PasteboardRefClearContents( CocoaPasteboardRef pb ) = NSInteger
```

```
PasteboardRefWriteObjects( CocoaPasteboardRef pb, CFArrayRef objects ) = Boolean
```

```
PasteboardRefSetStringForType( CocoaPasteboardRef pb, CFStringRef string, CFStringRef type ) = Boolean
```

Reading data

```
PasteboardRefItems( CocoaPasteboardRef pb ) = CFArrayRef
```

```
PasteboardRefPropertyListForType( CocoaPasteboardRef pb, CFStringRef type ) = CFTypeRef
```

```
PasteboardRefStringForType( CocoaPasteboardRef pb, CFStringRef type ) = CFStringRef
```

Validating contents

```
PasteboardRefTypes( CocoaPasteboardRef pb ) = CFArrayRef
```

Pasteboard info

```
PasteboardRefName( CocoaPasteboardRef pb ) = CFStringRef
```

```
PasteboardRefChangeCount( CocoaPasteboardRef pb ) = NSInteger
```

• Convenience functions - these use the general pasteboard

Writing data

```
PasteboardClearContents = NSInteger
```

```
PasteboardWriteObjects( CFArrayRef objects ) = Boolean
```

```
PasteboardSetStringForType( CFStringRef string, CFStringRef type ) = Boolean
```

Reading data

```
PasteboardItems = CFArrayRef
```

```
PasteboardPropertyListForType( CFStringRef type ) = CFTypeRef
```

```
PasteboardStringForType( CFStringRef type ) = CFStringRef
```

Validating contents

```
PasteboardTypes = CFArrayRef
```

Pasteboard info

```
PasteboardName = CFStringRef
```

```
PasteboardChangeCount = NSInteger
```

### Apple documentation

[NSPasteboard](#)



---

### PathUtilities

#### Functions

```
NSUserName = CFStringRef
NSFullUserName = CFStringRef
NSHomeDirectory = CFStringRef
NSHomeDirectoryForUser( CFStringRef userName ) = CFStringRef
NSTemporaryDirectory = CFStringRef
NSOpenStepRootDirectory = CFStringRef
```



## PDFDocument

### Functions

Init

```
PDFDocumentWithURL( CFURLRef url ) = PDFDocumentRef
PDFDocumentWithData( CFDataRef dta ) = PDFDocumentRef
PDFDocumentInit = PDFDocumentRef// autoreleased
```

Read and write

-- Read operations --

--- Info ---

```
PDFDocumentURL( PDFDocumentRef ref ) = CFURLRef
PDFDocumentMajorVersion( PDFDocumentRef ref ) = NSInteger
PDFDocumentMinorVersion( PDFDocumentRef ref ) = NSInteger
PDFDocumentString( PDFDocumentRef ref ) = CFStringRef
PDFDocumentOutlineItemForSelection( PDFDocumentRef docRef, PDFSelectionRef selRef ) = PDFOutlineRef
PDFDocumentOutlineRoot( PDFDocumentRef ref ) = PDFOutlineRef
PDFDocumentSetOutlineRoot( PDFDocumentRef docRef, PDFOutlineRef olRef )
PDFDocumentAttributes( PDFDocumentRef ref ) = CFDictionaryRef
PDFDocumentSetAttributes( PDFDocumentRef ref, CFDictionaryRef attributes )
PDFDocumentCGPDFRef( PDFDocumentRef ref ) = CGPDFDocumentRef
```

--- Security ---

```
PDFDocumentIsEncrypted( PDFDocumentRef ref ) = Boolean
PDFDocumentIsLocked( PDFDocumentRef ref ) = Boolean
PDFDocumentUnlockWithPassword( PDFDocumentRef ref, CFStringRef password ) = Boolean
PDFDocumentPermissionsStatus( PDFDocumentRef ref ) = PDFDocumentPermissions
```

---- Permissions properties ----

```
PDFDocumentAllowsCopying( PDFDocumentRef ref ) = Boolean
PDFDocumentAllowsPrinting( PDFDocumentRef ref ) = Boolean
PDFDocumentAllowsCommenting( PDFDocumentRef ref ) = BooleanmacOS 10.13+
PDFDocumentAllowsContentAccessibility( PDFDocumentRef ref ) = BooleanmacOS 10.13+
PDFDocumentAllowsDocumentAssembly( PDFDocumentRef ref ) = BooleanmacOS 10.13+
PDFDocumentAllowsDocumentChanges( PDFDocumentRef ref ) = BooleanmacOS 10.13+
PDFDocumentAllowsFormFieldEntry( PDFDocumentRef ref ) = BooleanmacOS 10.13+
```

--- Selections and searches ---

```
PDFDocumentSelectionFromPageAtIndex( PDFDocumentRef ref, PDFPageRef startPage, NSUInteger startCharIndex, PDFPageRef
endPage, NSUInteger endCharIndex ) = PDFSelectionRef
PDFDocumentSelectionFromPageAtPoint( PDFDocumentRef ref, PDFPageRef startPage, CGPoint startPt, PDFPageRef endPage,
CGPoint endPt ) = PDFSelectionRef
PDFDocumentSelectionForEntireDocument( PDFDocumentRef ref ) = PDFSelectionRef
```

---- Search options ----

```
PDFDocumentFindString( PDFDocumentRef ref, CFStringRef string, NSStringCompareOptions options ) = CFArrayRefarray of
selections
PDFDocumentFindStringFromSelection( PDFDocumentRef ref, CFStringRef string, PDFSelectionRef selection,
NSStringCompareOptions options ) = PDFSelectionRef
PDFDocumentIsFinding( PDFDocumentRef ref ) = Boolean
PDFDocumentCancelFindString( PDFDocumentRef ref )
```

--- Pages ---

```
PDFDocumentPageCount( PDFDocumentRef ref ) = NSUInteger
PDFDocumentPageAtIndex( PDFDocumentRef ref, NSUInteger index ) = PDFPageRef
PDFDocumentIndexForPage( PDFDocumentRef ref, PDFPageRef pg ) = NSUInteger
PDFDocumentInsertPageAtIndex( PDFDocumentRef ref, PDFPageRef pg, NSUInteger index )
PDFDocumentRemovePageAtIndex( PDFDocumentRef ref, NSUInteger index )
PDFDocumentExchangePageAtIndex( PDFDocumentRef ref, NSUInteger index1, NSUInteger index2 )
PDFDocumentPageClass( PDFDocumentRef ref ) = ClassRef
```

-- Write operations --

--- Data ---

```
PDFDocumentWriteToURL( PDFDocumentRef ref, CFURLRef url ) = Boolean
PDFDocumentWriteToURLWithOptions( PDFDocumentRef ref, CFURLRef url, CFDictionaryRef options ) = Booleanoptions is a
dictionary of PDFDocumentWriteOptions
```

---- Data representations ----

```
PDFDocumentDataRepresentation( PDFDocumentRef ref ) = CFDataRef
PDFDocumentDataRepresentationWithOptions( PDFDocumentRef ref, CFDictionaryRef options ) = CFDataRef
```

--- Printing ---

```
PDFDocumentPrintOperationForPrintInfo( PDFDocumentRef ref, PrintInfoRef info, PDFPrintScalingMode scalingMode,  
Boolean autorotate ) = PrintOperationRef
```

## Apple documentation

[PDFDocument](#)



---

## PDFInfo

### Functions

Init

```
PDFInfoInit = PDFInfoRef// autoreleased
```

Methods

```
PDFInfoURL( PDFInfoRef ref ) = CFURLRef
```

```
PDFInfoSetURL( PDFInfoRef ref, CFURLRef url )
```

```
PDFInfoIsFileExtensionHidden( PDFInfoRef ref ) = Boolean
```

```
PDFInfoSetFileExtensionHidden( PDFInfoRef ref, Boolean flag )
```

```
PDFInfoTagNames( PDFInfoRef ref ) = CFArrayRef
```

```
PDFInfoSetTagNames( PDFInfoRef ref, CFArrayRef names )
```

```
PDFInfoOrientation( PDFInfoRef ref ) = NSPaperOrientation
```

```
PDFInfoSetOrientation( PDFInfoRef ref, NSPaperOrientation orientation )
```

```
PDFInfoPaperSize( PDFInfoRef ref ) = CGSize
```

```
PDFInfoSetPaperSize( PDFInfoRef ref, CGSize size )
```

```
PDFInfoAttributes( PDFInfoRef ref ) = CFMutableDictionaryRef// dictionary of PrintInfo attributes
```

### Apple documentation

[NSPDFInfo](#)



---

## PDFPanel

### Functions

Create

```
PDFPanelInit = PDFPanelRef // autoreleased
```

Contents

```
PDFPanelOptions( PDFPanelRef ref ) = NSPDFPanelOptions
```

```
PDFPanelSetOptions( PDFPanelRef ref, NSPDFPanelOptions options )
```

```
PDFPanelDefaultFileName( PDFPanelRef ref ) = CFStringRef
```

```
PDFPanelSetDefaultFileName( PDFPanelRef ref, CFStringRef fileName )
```

Display

```
PDFPanelBeginSheet( PDFPanelRef ref, PDFInfoRef info, NSInteger wndTag, ptr callback, ptr userData )
```

### Apple documentation

[NSPDFPanel](#)





## PDFThumbnailView

statement

### Syntax

```
pdfthumbnailview tag, rect, pdfViewTag, wndTag
```

### Description

The **pdfthumbnailview** statement puts a new pdfthumbnailview in the current output cocoa window, or alters an existing pdfthumbnailview's characteristics.

### Parameters

<i>tag</i>	A number to identify the thumbnailview. A negative tag hides the view.
<i>rect</i>	Origin and size of the view in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>pdfViewTag</i>	The tag of a linked pdfview.
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Functions

```
PDFThumbnailViewWithTag( NSInteger tag ) = PDFThumbnailViewRef
```

PDFView

```
PDFThumbnailViewPDFView( NSInteger tag ) = PDFViewRef
```

```
PDFThumbnailViewSetPDFView( NSInteger thumbnailTag, NSInteger pdfTag )
```

Size

```
PDFThumbnailViewThumbnailSize( NSInteger tag ) = CGSize
```

```
PDFThumbnailViewSetThumbnailSize( NSInteger tag, CGSize size )
```

Display

```
PDFThumbnailViewMaximumNumberOfColumns( NSInteger tag ) = NSUInteger
```

```
PDFThumbnailViewSetMaximumNumberOfColumns( NSInteger tag, NSUInteger columns )
```

```
PDFThumbnailViewLabelFont( NSInteger tag ) = FontRef
```

```
PDFThumbnailViewSetLabelFont( NSInteger tag, FontRef font )
```

```
PDFThumbnailViewBackgroundColor( NSInteger tag ) = ColorRef
```

```
PDFThumbnailViewSetBackgroundColor( NSInteger tag, ColorRef col )
```

Behaviour

```
PDFThumbnailViewAllowsDragging( NSInteger tag ) = Boolean
```

```
PDFThumbnailViewSetAllowsDragging( NSInteger tag, Boolean flag )
```

```
PDFThumbnailViewAllowsMultipleSelection( NSInteger tag ) = Boolean
```

```
PDFThumbnailViewSetMultipleSelection( NSInteger tag, Boolean flag )
```

```
PDFThumbnailViewSelectedPages( NSInteger tag ) = CFArrayRef
```

### Apple documentation

[PDFThumbnailView](#)



## PDFView

statement

### Syntax

**pdfview** *tag, rect, pdfDocRef, wndTag*

### Description

The **pdfview** statement puts a new pdfview in the current output cocoa window, or alters an existing pdfview's characteristics.

### Parameters

<i>tag</i>	A number to identify the pdfview. A negative tag hides the view.
<i>rect</i>	Origin and size of the view in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>pdfDocRef</i>	The pdfdocument ref.
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Functions

```
PDFViewWithTag( NSInteger tag ) = PDFViewRef
```

Document

```
PDFViewDocument( NSInteger tag ) = PDFDocumentRef
```

```
PDFViewSetDocument( NSInteger tag, PDFDocumentRef doc )
```

Configurations

-- Display mode --

```
PDFViewDisplayMode( NSInteger tag ) = PDFDisplayMode
```

```
PDFViewSetDisplayMode( NSInteger tag, PDFDisplayMode displayMode )
```

-- Additional display configurations --

--- Display box ---

```
PDFViewDisplayBox( NSInteger tag ) = PDFDisplayBox
```

```
PDFViewSetDisplayBox( NSInteger tag, PDFDisplayBox displayBox )
```

--- Page breaks ---

```
PDFViewDisplaysPageBreaks( NSInteger tag ) = Boolean // macOS 10.13+
```

```
PDFViewSetDisplaysPageBreaks( NSInteger tag, Boolean flag ) // macOS 10.13+
```

```
PDFViewPageBreakMargins( NSInteger tag ) = NSEdgeInsets // macOS 10.13+
```

```
PDFViewSetPageBreakMargins( NSInteger tag, NSEdgeInsets insets ) // macOS 10.13+
```

--- Display direction ---

```
PDFViewDisplayDirection( NSInteger tag ) = PDFDisplayDirection // macOS 10.13+
```

```
PDFViewSetDisplayDirection( NSInteger tag, PDFDisplayDirection direction ) // macOS 10.13+
```

```
PDFViewDisplaysRTL( NSInteger tag ) = Boolean // macOS 10.13+
```

```
PDFViewSetDisplaysRTL( NSInteger tag, Boolean flag ) // macOS 10.13+
```

-- Book display --

--- Display as book ---

```
PDFViewDisplaysAsBook( NSInteger tag ) = Boolean
```

```
PDFViewSetDisplaysAsBook( NSInteger tag, Boolean flag )
```

-- Graphics properties --

--- Background color ---

```
PDFViewBackgroundColor( NSInteger tag ) = ColorRef
```

```
PDFViewSetBackgroundColor( NSInteger tag, ColorRef col )
```

- Antialiasing -

```
PDFViewInterpolationQuality( NSInteger tag ) = PDFInterpolationQuality
```

-- Scaling the view --

```
PDFViewScaleFactor( NSInteger tag ) = CGFloat
```

```
PDFViewSetScaleFactor( NSInteger tag, CGFloat factor )
```

```
PDFViewScaleFactorForSizeToFit( NSInteger tag ) = CGFloat // macOS 10.13+
```

```
PDFViewMaxScaleFactor( NSInteger tag ) = CGFloat // macOS 10.13+
```

```
PDFViewSetMaxScaleFactor( NSInteger tag, CGFloat factor ) // macOS 10.13+
```

```

PDFViewMinScaleFactor( NSInteger tag ) = CGFloat // macOS 10.13+
PDFViewSetMinScaleFactor( NSInteger tag, CGFloat factor ) // macOS 10.13+
PDFViewAutoScales( NSInteger tag ) = Boolean
PDFViewSetAutoScales( NSInteger tag, Boolean flag )
PDFViewRowSizeForPage( NSInteger tag, PDFPageRef pg ) = CGSize

— Zoom operations ---
PDFViewZoomIn( NSInteger tag )
PDFViewCanZoomIn( NSInteger tag ) = Boolean
PDFViewZoomOut( NSInteger tag )
PDFViewCanZoomOut( NSInteger tag ) = Boolean

-- Rendering and printing the view --
PDFViewPrintWithInfo( NSInteger tag, PrintInfoRef printInfo, Boolean autoRotate )
PDFViewPrintWithInfoPageScaling( NSInteger tag, PrintInfoRef printInfo, Boolean autoRotate, PDFPrintScalingMode
scale )

-- Specializing the view
PDFViewDocumentView( NSInteger tag ) = ViewRef
PDFViewLayoutDocumentView( NSInteger tag )

-- Draw operations --
PDFViewDrawPageToContext( NSInteger tag, PDFPageRef pg, CGContextRef context ) // macOS 10.12+
PDFViewDrawPagePostToContext( NSInteger tag, PDFPageRef pg, CGContextRef context ) // macOS 10.12+

-- Document interactions --
--- Selections ---
PDFViewCurrentSelection( NSInteger tag ) = PDFSelectionRef
PDFViewSetCurrentSelection( NSInteger tag, PDFSelectionRef selection, Boolean animate )
PDFViewSelectAll( NSInteger tag )
PDFViewClearSelection( NSInteger tag )
PDFViewCopy( NSInteger tag )
PDFViewScrollSelectionToVisible( NSInteger tag )
PDFViewHighlightedSelections( NSInteger tag ) = CFArrayRef

--- Annotation actions ---
PDFViewAnnotationsChangedOnPage( NSInteger tag, PDFPageRef pg )

--- Link annotations ---
PDFViewEnableDataDetectors( NSInteger tag ) = Boolean

--- Convert page and view points ---
PDFViewPageForPoint( NSInteger tag, CGPoint pt, Boolean nearest ) = PDFPageRef
PDFViewConvertPointToPage( NSInteger tag, CGPoint pt, PDFPageRef pg ) = CGPoint
PDFViewConvertRectToPage( NSInteger tag, CGRect r ) = PDFPageRef
PDFViewConvertPointFromPage( NSInteger tag, CGPoint pt, PDFPageRef pg ) = CGPoint
PDFViewConvertRectFromPage( NSInteger tag, CGRect r, PDFPageRef pg ) = CGRect

--- Mouse position and events ---
PDFViewAreaOfInterestForMouse( NSInteger tag, CocoaEventRef evnt ) = PDFAreaOfInterest
PDFViewAreaOfInterestForPoint( NSInteger tag, CGPoint pt ) = PDFAreaOfInterest // macOS 10.10.2+
PDFViewSetCursorForAreaOfInterest( NSInteger tag, PDFAreaOfInterest area )
PDFViewPerformAction( NSInteger tag, PDFActionRef action )

--- Drag operations
PDFViewAcceptsDraggedFiles( NSInteger tag ) = Boolean // macOS 10.13+
PDFViewSetAcceptsDraggedFiles( NSInteger tag, Boolean flag ) // macOS 10.13+

— Navigating within document --
PDFViewCurrentPage( NSInteger tag ) = PDFPageRef
PDFViewCurrentDestination( NSInteger tag ) = PDFDestinationRef
PDFViewVisiblePages( NSInteger tag ) = CFArrayRef

-- Navigation --
PDFViewCanGoBack( NSInteger tag ) = Boolean
PDFViewCanGoForward( NSInteger tag ) = Boolean
PDFViewCanGoToFirstPage( NSInteger tag ) = Boolean
PDFViewCanGoToLastPage( NSInteger tag ) = Boolean
PDFViewCanGoToNextPage( NSInteger tag ) = Boolean
PDFViewCanGoToPreviousPage( NSInteger tag ) = Boolean

PDFViewGoBack( NSInteger tag )
PDFViewGoForward( NSInteger tag )
PDFViewGoToFirstPage( NSInteger tag )
PDFViewGoToLastPage( NSInteger tag )
PDFViewGoToNextPage( NSInteger tag )
PDFViewGoToPreviousPage( NSInteger tag )
PDFViewGoToPage( NSInteger tag, PDFPageRef pg )
PDFViewGoToDestination( NSInteger tag, PDFDestinationRef destination )
PDFViewGoToSelection( NSInteger tag, PDFSelectionRef selection )
PDFViewGoToRectOnPage( NSInteger tag, CGRect r, PDFPageRef pg )

```

- Convenience -

```
PDFViewCurrentPageIndex( NSInteger tag ) = NSUInteger  
PDFViewGoToPageAtIndex( NSInteger tag, NSUInteger index )
```

## **Apple documentation**

[PDFView](#)



---

### Pipe

#### Functions

Init

```
PipeInit = PipeRef// autoreleased
```

File handles

```
PipeFileHandleForReading( PipeRef p ) = FileHandleRef
```

```
PipeFileHandleForWriting( PipeRef p ) = FileHandleRef
```

#### Apple documentation

[NSPipe](#)



---

## PointerArray

### Functions

#### Create

```
PointerArrayWithOptions( NSPointerFunctionsOptions options ) = PointerArrayRef
PointerArrayWithPointerFunctions( PointerFunctionsRef pf ) = PointerArrayRef
PointerArrayStrongObjects = PointerArrayRef// macOS 10.8+
PointerArrayWeakObjects = PointerArrayRef// macOS 10.8+
```

#### Manage

```
PointerArrayCount( PointerArrayRef pa ) = NSUInteger
PointerArrayAllObjects( PointerArrayRef pa ) = CFArrayRef
PointerArrayPointerAtIndex( PointerArrayRef pa, NSUInteger index ) = ptr
PointerArrayAddPointer( PointerArrayRef pa, ptr p )
PointerArrayRemovePointerAtIndex( PointerArrayRef pa, NSUInteger index )
PointerArrayInsertPointerAtIndex( PointerArrayRef pa, ptr p, NSUInteger index )
PointerArrayReplacePointerAtIndex( PointerArrayRef pa, ptr p, NSUInteger index )
PointerArrayCompact( PointerArrayRef pa )
```

#### Pointer functions

```
PointerArrayPointerFunctions( PointerArrayRef pa ) = PointerFunctionsRef
```

### Apple documentation

[NSPointerArray](#)



---

## PointerFunctions

### Functions

Create  
`PointerFunctionsWithOptions( NSPointerFunctionsOptions options ) = PointerFunctionsRef`

Personality functions

```
PointerFunctionsHashFunction( PointerFunctionsRef pf ) = ptr
PointerFunctionsSetHashFunction( PointerFunctionsRef pf, ptr fnAddress )
PointerFunctionsIsEqualFunction( PointerFunctionsRef pf ) = ptr
PointerFunctionsSetIsEqualFunction( PointerFunctionsRef pf, ptr fnAddress )
PointerFunctionsSizeFunction( PointerFunctionsRef pf ) = ptr
PointerFunctionsSetSizeFunction( PointerFunctionsRef pf, ptr fnAddress )
PointerFunctionsDescriptionFunction( PointerFunctionsRef pf ) = ptr
PointerFunctionsSetDescriptionFunction( PointerFunctionsRef pf, ptr fnAddress )
```

Memory configuration

```
PointerFunctionsAcquireFunction( PointerFunctionsRef pf ) = ptr
PointerFunctionsSetAcquireFunction( PointerFunctionsRef pf, ptr fnAddress )
PointerFunctionsRelinquishFunction( PointerFunctionsRef pf ) = ptr
PointerFunctionsSetRelinquishFunction( PointerFunctionsRef pf, ptr fnAddress )
```

### Apple documentation

[NSPointerFunctions](#)



## Popover

statement

### Syntax

**popover** *tag, rect, behavior, animates*

### Description

The **popover** statement creates a new popover.

### Parameters

<i>tag</i>	Unique number to identify the popover. The tag value cannot be the same value as an existing Carbon or cocoa window. A negative value closes the popover.
<i>rect</i>	Origin and size of the popover's content area. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <code>CGRect</code> value
<i>behavior</i>	How the popover behaves. <code>NSPopoverBehaviorApplicationDefined</code> (default) <code>NSPopoverBehaviorTransient</code> <code>NSPopoverBehaviorSemitransient</code>
<i>animates</i>	Specifies if the popover is to be animated (default = <code>_false</code> ).

### Dialog Events

Event	Description
<code>_popoverDetachableWindow</code>	This event requests a detachable window tag value. Submit the window tag with <b>DialogEventSetLong</b> ( wndTag ).
<code>_popoverShouldClose</code>	Allows override of close request. Call <b>DialogEventSetBool</b> ( <code>_false</code> ) to cancel close of the popover.
<code>_popoverWillShow</code>	The popover will show.
<code>_popoverDidShow</code>	The popover has been shown.
<code>_popoverWillClose</code>	The popover is about to close.
<code>_popoverDidClose</code>	The popover did close.
<code>_popoverDidDetach</code>	The popover did detach (macOS 10.10)
<code>_popoverShouldDetach</code>	Allows override of detach request. (macOS 10.10). Call <b>DialogEventSetBool</b> ( <code>_false</code> ) to cancel detach of the popover.

### Functions

```
PopoverWithTag( NSInteger tag ) = PopoverRef
PopoverExists( NSInteger tag ) = Boolean
```

```
Init
PopoverInit( NSInteger tag, CGRect r ) = PopoverRef// autoreleased
```

```
Position and size
PopoverBehavior( NSInteger tag ) = NSPopoverBehavior
PopoverSetBehavior( NSInteger tag, NSPopoverBehavior behavior )
PopoverPositioningRect( NSInteger tag ) = CGRect
PopoverSetPositioningRect( NSInteger tag, CGRect rect )
```

```
Appearance
PopoverAppearance( NSInteger tag ) = AppearanceRef// macOS 10.10+
PopoverSetAppearance( NSInteger tag, AppearanceRef app )// macOS 10.10+
PopoverEffectiveAppearance( NSInteger tag ) = AppearanceRef// macOS 10.10+
PopoverAnimates( NSInteger tag ) = Boolean
PopoverSetAnimates( NSInteger tag, Boolean flag )
PopoverContentSize( NSInteger tag ) = CGSize
```



`PopoverSetContentSize( NSInteger tag, CGSize size )`  
`PopoverIsShown = Boolean`  
`PopoverIsDetached( NSInteger tag ) = Boolean` macOS 10.10+

Closing  
`PopoverPerformClose( NSInteger tag )`  
`PopoverClose`

Convenience  
`PopoverAddSubview`  
`PopoverShow`  
`PopoverSetDetachableWindow( NSInteger popoverTag, NSInteger wndTag )`

## See Also

See [NibPopover](#)

## Apple documentation

[NSPopover](#)



## PopUpButton

statement

### Syntax

**popupbutton** *tag, enabled, index, items, rect, pullsDown, wndTag*

### Description

The **popupbutton** statement puts a new popupbutton in the current output cocoa window, or alters an existing popupbutton's characteristics.

### Parameters

<i>tag</i>	A number to identify the popupbutton. A negative tag hides the popupbutton.
<i>enabled</i>	Enables or disables the popupbutton.
<i>index</i>	The popupbutton's selected item index (default = 0).
<i>items</i>	A CFArray or semicolon-delimited list of item titles. An item name containing a single dash (-) will be converted to menu separator.
<i>rect</i>	Origin and size of the popupbutton in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>pullsDown</i>	Specify whether the popupbutton displays a pull-down menu; otherwise it displays a pop-up menu (default = <code>_false</code> ).
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

[\\_btnClick](#)

### Functions

[PopUpButtonWithTag\( NSInteger tag \) = PopUpButtonRef](#)

[PopUpButtonExists\( NSInteger tag \) = Boolean](#)

Init

[PopUpButtonInit\( NSInteger tag, CGRect r \) = PopUpButtonRef// autoreleased](#)

Type of menu

[PopUpButtonPullsDown\( NSInteger tag \) = Boolean](#)

[PopUpButtonSetPullsDown\( NSInteger tag, Boolean flag \)](#)

[PopUpButtonAutoenablesItems\( NSInteger tag \) = Boolean](#)

[PopUpButtonSetAutoenablesItems\( NSInteger tag, Boolean flag \)](#)

Inserting and deleting items

[PopUpButtonAddItemWithTitle\( NSInteger tag, CFStringRef title \)](#)

[PopUpButtonAddItemsWithTitles\( NSInteger tag, CFArrayRef titles \)](#)

[PopUpButtonInsertItemWithTitle\( NSInteger tag, CFStringRef title, NSInteger index \)](#)

[PopUpButtonRemoveAllItems\( NSInteger tag \)](#)

[PopUpButtonRemoveItemWithTitle\( NSInteger tag, CFStringRef title \)](#)

[PopUpButtonRemoveItemAtIndex\( NSInteger tag, NSInteger index \)](#)

User's selection

[PopUpButtonTitleOfSelectedItem\( NSInteger tag \) = CFStringRef](#)

[PopUpButtonIndexOfSelectedItem\( NSInteger tag \) = NSInteger](#)

Setting current selection

[PopUpButtonSelectItemAtIndex\( NSInteger tag, NSInteger index \)](#)

[PopUpButtonSelectItemWithTag\( NSInteger popUpTag, NSInteger itemTag \)](#)

[PopUpButtonSelectItemWithTitle\( NSInteger tag, CFStringRef title \)](#)

Menu items

[PopUpButtonMenu\( NSInteger tag \) = CocoaMenuRef](#)

[PopUpButtonSetMenu\( NSInteger tag, CocoaMenuRef menu \)](#)

[PopUpButtonMenuItemIndex\( NSInteger tag \) = NSInteger](#)

[PopUpButtonSetMenuItemIndex\( NSInteger tag, NSInteger menuItemIndex \)](#)

```
PopupMenuNumberOfItems( NSInteger tag ) = NSInteger
PopupMenuItemArray( NSInteger tag ) = CFArrayRef
PopupMenuItemTitleAtIndex( NSInteger tag, NSInteger index ) = CFStringRef
PopupMenuItemTitles( NSInteger tag ) = CFArrayRef
```

Indices of menu items

```
PopupMenuIndexOfItemWithTag( NSInteger popUpTag, NSInteger itemTag ) = NSInteger
PopupMenuIndexOfItemWithTitle( NSInteger tag, CFStringRef title ) = NSInteger
```

Preferred edge

```
PopupMenuPreferredEdge( NSInteger tag ) = CGRectEdge
PopupMenuSetPreferredEdge( NSInteger tag, CGRectEdge edge )
```

Title

```
PopupMenuSetTitle( NSInteger tag, CFStringRef title )
```

State

```
PopupMenuSynchronizeTitleAndSelectedItem( NSInteger tag )
```

Cell methods

```
PopupMenuUsesItemFromMenu( NSInteger tag ) = Boolean
PopupMenuSetUsesItemFromMenu( NSInteger tag, Boolean flag )
PopupMenuAltersStateOfSelectedItem( NSInteger tag ) = Boolean
PopupMenuSetAltersStateOfSelectedItem( NSInteger tag, Boolean flag )
PopupMenuArrowPosition( NSInteger tag ) = NSPopupMenuArrowPosition
PopupMenuSetArrowPosition( NSInteger tag, NSPopupMenuArrowPosition position )
```

Convenience

```
PopupMenuItemSetEnabled( NSInteger tag, NSInteger index, Boolean flag )
```

## See Also

[Control](#), [View](#)

## Apple documentation

[NSPopupMenu](#)



---

## Predicate

### Functions

Create

```
PredicateWithFormat( CFStringRef format, ... ) = PredicateRef
PredicateWithFormatAndArgumentArray( CFStringRef format, CFArrayRef arguments ) = PredicateRef
PredicateWithSubstitutionVariables( PredicateRef pred, CFDictionaryRef variables ) = PredicateRef
PredicateWithValue( Boolean value ) = PredicateRef
PredicateFromMetadataQueryString( CFStringRef string ) = PredicateRef// macOS 10.9+
```

Evaluate

```
PredicateEvaluateWithObject( PredicateRef pred, CTypeRef obj ) = Boolean
PredicateEvaluateWithObjectAndSubstitutionVariables( PredicateRef pred, CTypeRef obj, CFDictionaryRef variables ) = Boolean
PredicateAllowEvaluation( PredicateRef pred )// macOS 10.9+
```

String representation

```
PredicateFormat( PredicateRef pred ) = CFStringRef
```

### Apple documentation

[NSPredicate](#)



---

## PressGestureRecognizer

### Functions

Init

```
PressGestureRecognizerInit( ptr callback, ptr userData ) = PressGestureRecognizerRef// autoreleased
```

Configure

```
PressGestureRecognizerButtonMask( PressGestureRecognizerRef ref ) = NSUInteger  
PressGestureRecognizerSetButtonMask( PressGestureRecognizerRef ref, NSUInteger mask )  
PressGestureRecognizerMinimumPressDuration( PressGestureRecognizerRef ref ) = CFTimeInterval  
PressGestureRecognizerSetMinimumPressDuration( PressGestureRecognizerRef ref, CFTimeInterval duration )  
PressGestureRecognizerAllowableMovement( PressGestureRecognizerRef ref ) = CGFloat  
PressGestureRecognizerSetAllowableMovement( PressGestureRecognizerRef ref, CGFloat movement )
```

Instance properties

```
PressGestureRecognizerNumberOfTouchesRequired( PressGestureRecognizerRef ref ) = NSInteger// macOS 10.12.2  
PressGestureRecognizerSetNumberOfTouchesRequired( PressGestureRecognizerRef ref, NSInteger touches )// macOS 10.12.2
```

### Apple documentation

[NSPressGestureRecognizer](#)



---

## PressureConfiguration

### Functions

Create

```
PressureConfigurationWithBehavior( NSPressureBehavior behavior ) = PressureConfigurationRef// macOS 10.11+  
PressureConfigurationSet( PressureConfigurationRef ref )// macOS 10.11+
```

Properties

```
PressureConfigurationBehavior( PressureConfigurationRef ref ) = NSPressureBehavior// macOS 10.11+
```

### Apple documentation

[NSPressureConfiguration](#)



---

## Printer

### Functions

Create

```
PrinterWithName( CFStringRef name ) = PrinterRef  
PrinterWithType( NSPrinterTypeName type ) = PrinterRef
```

General info

```
PrinterNames = CFArrayRef  
PrinterTypes = CFArrayRef
```

Attributes

```
PrinterName( PrinterRef printer ) = CFStringRef  
PrinterType( PrinterRef printer ) = NSPrinterTypeName
```

Specific info

```
PrinterPageSizeForPaper( PrinterRef printer, NSPrinterPaperName paperName ) = CGSize  
PrinterLanguageLevel( PrinterRef printer ) = NSInteger
```

Query tables

```
PrinterDeviceDescription( PrinterRef printer ) = CFDictionaryRef
```

### Apple documentation

[NSPrinter](#)



---

## PrintInfo

### Functions

Shared print info

```
PrintInfoShared = PrintInfoRef
```

Printing rectangle

```
PrintInfoBottomMargin = CGFloat
PrintInfoSetBottomMargin( CGFloat value )
PrintInfoImageablePageBounds = CGRect
PrintInfoLeftMargin = CGFloat
PrintInfoSetLeftMargin( CGFloat value )
PrintInfoOrientation = NSPaperOrientation
PrintInfoSetOrientation( NSPaperOrientation orientation )
PrintInfoPaperName = NSPrinterPaperName
PrintInfoSetPaperName( NSPrinterPaperName string )
PrintInfoLocalizedPaperName = CFStringRef
PrintInfoPaperSize = CGSize
PrintInfoSetPaperSize( CGSize size )
PrintInfoRightMargin = CGFloat
PrintInfoSetRightMargin( CGFloat value )
PrintInfoTopMargin = CGFloat
PrintInfoSetTopMargin( CGFloat value )
```

Pagination

```
PrintInfoHorizontalPagination = NSPrintingPaginationMode
PrintInfoSetHorizontalPagination( NSPrintingPaginationMode value )
PrintInfoVerticalPagination = NSPrintingPaginationMode
PrintInfoSetVerticalPagination( NSPrintingPaginationMode value )
```

Positioning

```
PrintInfoIsHorizontallyCentered = Boolean
PrintInfoSetHorizontallyCentered( Boolean flag )
PrintInfoIsVerticallyCentered = Boolean
PrintInfoSetVerticallyCentered( Boolean flag )
```

Printer

```
PrintInfoPrinter = PrinterRef
PrintInfoSetPrinter( PrinterRef printer )
```

Controlling printing

```
PrintInfoJobDisposition = NSPrintJobDispositionValue
PrintInfoSetJobDisposition( NSPrintJobDispositionValue value )
PrintInfoSetUpPrintOperationDefaultValues
```

Print info dictionary

```
PrintInfoDictionary = CFDictionaryRef
```

Print settings convenience functions

```
PrintInfoIsSelectionOnly = Boolean
PrintInfoSetSelectionOnly( Boolean flag )
PrintInfoScalingFactor = CGFloat
PrintInfoSetScalingFactor( CGFloat value )
```

Core print info

```
PrintInfoPrintSettings = CFMutableDictionaryRef
```

Instance methods

```
PrintInfoTakeSettingsFromPDFInfo( PDFInfoRef info )// macOS 10.9+
```

### Apple documentation

[NSPrintInfo](#)





## PrintOperation

### Functions

#### Create

```
PrintOperationEPSWithViewToData( NSInteger viewTag, CGRect rect, CFMutableDataRef data ) = PrintOperationRef
PrintOperationEPSWithViewToDataPrintInfo( NSInteger viewTag, CGRect rect, CFMutableDataRef data, PrintInfoRef
printInfo ) = PrintOperationRef
PrintOperationEPSWithViewToPath( NSInteger viewTag, CGRect rect, CFStringRef path, PrintInfoRef printInfo ) =
PrintOperationRef
PrintOperationPDFWithViewToData( NSInteger viewTag, CGRect rect, CFMutableDataRef data ) = PrintOperationRef
PrintOperationPDFWithViewToDataPrintInfo( NSInteger viewTag, CGRect rect, CFMutableDataRef data, PrintInfoRef
printInfo ) = PrintOperationRef
PrintOperationPDFWithViewToPath( NSInteger viewTag, CGRect rect, CFStringRef path, PrintInfoRef printInfo ) =
PrintOperationRef
PrintOperationWithView( NSInteger viewTag ) = PrintOperationRef
PrintOperationWithViewPrintInfo( NSInteger viewTag, PrintInfoRef printInfo ) = PrintOperationRef
```

#### Current print operation

```
PrintOperationCurrent = PrintOperationRef
```

#### Type

```
PrintOperationIsCopying( PrintOperationRef printOperation ) = Boolean
```

#### Modify print info

```
PrintOperationPrintInfo( PrintOperationRef printOperation ) = PrintInfoRef
PrintOperationSetPrintInfo( PrintOperationRef printOperation, PrintInfoRef printInfo )
```

#### NSView object

```
PrintOperationView( PrintOperationRef printOperation ) = NSInteger
```

#### Printing quality

```
PrintOperationPreferredRenderingQuality( PrintOperationRef printOperation ) = NSPrintRenderingQuality
```

#### Running

```
PrintOperationRun( PrintOperationRef printOperation ) = Boolean
PrintOperationRunModal( PrintOperationRef printOperation, NSInteger wndTag, ptr callback, ptr userData )
PrintOperationDeliverResult( PrintOperationRef printOperation ) = Boolean
```

#### Modify user interface

```
PrintOperationShowsPrintPanel( PrintOperationRef printOperation ) = Boolean
PrintOperationSetShowsPrintPanel( PrintOperationRef printOperation, Boolean flag )
PrintOperationShowsProgressPanel( PrintOperationRef printOperation ) = Boolean
PrintOperationSetShowsProgressPanel( PrintOperationRef printOperation, Boolean flag )
PrintOperationJobTitle( PrintOperationRef printOperation ) = CFStringRef
PrintOperationSetJobTitle( PrintOperationRef printOperation, CFStringRef title )
PrintOperationPrintPanel( PrintOperationRef printOperation ) = PrintPanelRef
PrintOperationSetPrintPanel( PrintOperationRef printOperation, PrintPanelRef printPanel )
```

#### Drawing context

```
PrintOperationContext( PrintOperationRef printOperation ) = GraphicsContextRef
PrintOperationCreateContext( PrintOperationRef printOperation ) = GraphicsContextRef
```

#### Page info

```
PrintOperationCurrentPage( PrintOperationRef printOperation ) = NSInteger
PrintOperationPageRange( PrintOperationRef printOperation ) = CFRange
PrintOperationPageOrder( PrintOperationRef printOperation ) = NSPrintingPageOrder
PrintOperationSetPageOrder( PrintOperationRef printOperation, NSPrintingPageOrder pageOrder )
```

#### Managing print-related threads

```
PrintOperationCanSpawnSeparateThread( PrintOperationRef printOperation ) = Boolean
PrintOperationSetCanSpawnSeparateThread( PrintOperationRef printOperation, Boolean flag )
```

### Apple documentation

[NSPrintOperation](#)



---

## PrintPanel

### Functions

Create

```
PrintPanelInit = PrintPanelRef// autoreleased
```

Customize

```
PrintPanelJobStyleHint( PrintPanelRef pp ) = NSPrintPanelJobStyleHint
PrintPanelSetJobStyleHint( PrintPanelRef pp, NSPrintPanelJobStyleHint hint )
PrintPanelOptions( PrintPanelRef pp ) = NSPrintPanelOptions
PrintPanelSetOptions( PrintPanelRef pp, NSPrintPanelOptions options )
PrintPanelDefaultButtonTitle( PrintPanelRef pp ) = CFStringRef
PrintPanelSetDefaultButtonTitle( PrintPanelRef pp, CFStringRef string )
PrintPanelHelpAnchor( PrintPanelRef pp ) = CFStringRef
PrintPanelSetHelpAnchor( PrintPanelRef pp, CFStringRef string )
```

Running

```
PrintPanelBeginSheet( PrintPanelRef pp, PrintInfoRef printInfo, NSInteger wndTag, ptr callback, ptr userData )
PrintPanelRunModal( PrintPanelRef pp ) = NSInteger
PrintPanelRunModalWithPrintInfo( PrintPanelRef pp, PrintInfoRef po ) = NSInteger
```

Communicating with print info

```
PrintPanelPrintInfo( PrintPanelRef pp ) = PrintInfoRef
```

### Apple documentation

[NSPrintPanel](#)



---

## ProcessInfo

### Functions

Get process info agent

```
ProcessInfoInit = ProcessInfoRef// autoreleased
```

Accessing process info

```
ProcessInfoArguments = CFArrayRef  
ProcessInfoEnvironment = CFDictionaryRef  
ProcessInfoGloballyUniqueString = CFStringRef  
ProcessInfoProcessIdentifier = long  
ProcessInfoProcessName = CFStringRef
```

User info

```
ProcessInfoUserName = CFStringRef// macOS 10.12+  
ProcessInfoFullUserName = CFStringRef// macOS 10.12+
```

Sudden app termination

```
ProcessInfoDisableSuddenTermination  
ProcessInfoEnableSuddenTermination
```

Control auto termination

```
ProcessInfoDisableAutomaticTermination( CFStringRef reason )  
ProcessInfoEnableAutomaticTermination( CFStringRef reason )  
ProcessInfoAutomaticTerminationSupportEnabled = Boolean
```

Host info

```
ProcessInfoHostName = CFStringRef  
ProcessInfoOperatingSystemVersionString = CFStringRef  
ProcessInfoOperatingSystemVersion = NSOperatingSystemVersion// macOS 10.10+  
ProcessInfoIsOperatingSystemAtLeastVersion( NSOperatingSystemVersion version) = Boolean// macOS 10.10+
```

Computer info

```
ProcessInfoProcessorCount = NSUInteger  
ProcessInfoActiveProcessorCount = NSUInteger  
ProcessInfoPhysicalMemory = UInt64  
ProcessInfoSystemUptime = CFTimeInterval
```

Activities

```
ProcessInfoBeginActivity( NSActivityOptions options, CFStringRef reason ) = ptr// macOS 10.9+  
ProcessInfoEndActivity( ptr obj )// macOS 10.9+
```

Thermal state

```
ProcessInfoThermalState = NSProcessInfoThermalState// macOS 10.10.3+
```

### Apple documentation

[NSProcessInfo](#)



## ProgressIndicator

statement

### Syntax

```
progressindicator tag, value, rect, min, max, wndTag
```

### Description

The **progressindicator** statement puts a new progressindicator in the current output cocoa window, or alters an existing progressindicator's characteristics.

### Parameters

<i>tag</i>	A number to identify the progressindicator. A negative tag hides the progressindicator.
<i>value</i>	The value that indicates the current extent of the progressindicator (default = 0).
<i>rect</i>	Origin and size of the progressindicator in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>min</i>	The minimum value of the progressindicator (default = 0).
<i>max</i>	The maximum value of the progressindicator (default = 100).
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Functions

```
ProgressIndicatorWithTag( NSInteger tag ) = ProgressIndicatorRef  
ProgressIndicatorExists( NSInteger tag ) = Boolean
```

#### Init

```
ProgressIndicatorInit( NSInteger tag, CGRect r ) = ProgressIndicatorRef// autoreleased
```

#### Animating

```
ProgressIndicatorStartAnimation( NSInteger tag )  
ProgressIndicatorStopAnimation( NSInteger tag )  
ProgressIndicatorUsesThreadedAnimation( NSInteger tag ) = Boolean  
ProgressIndicatorSetUsesThreadedAnimation( NSInteger tag, Boolean flag )
```

#### Advancing the progress bar

```
ProgressIndicatorIncrementBy( NSInteger tag, double delta )  
ProgressIndicatorDoubleValue( NSInteger tag ) = double  
ProgressIndicatorSetDoubleValue( NSInteger tag, double value )  
ProgressIndicatorMinValue( NSInteger tag ) = double  
ProgressIndicatorSetMinValue( NSInteger tag, double value )  
ProgressIndicatorMaxValue( NSInteger tag ) = double  
ProgressIndicatorSetMaxValue( NSInteger tag, double value )
```

#### Appearance

```
ProgressIndicatorControlSize( NSInteger tag ) = NSControlSize  
ProgressIndicatorSetControlSize( NSInteger tag, NSControlSize size )  
ProgressIndicatorControlTint( NSInteger tag ) = NSControlTint  
ProgressIndicatorSetControlTint( NSInteger tag, NSControlTint tint )  
ProgressIndicatorIsBezeled( NSInteger tag ) = Boolean  
ProgressIndicatorSetBezeled( NSInteger tag, Boolean flag )  
ProgressIndicatorIsIndeterminate( NSInteger tag ) = Boolean  
ProgressIndicatorSetIndeterminate( NSInteger tag, Boolean flag )  
ProgressIndicatorStyle( NSInteger tag ) = NSUInteger  
ProgressIndicatorSetStyle( NSInteger tag, NSUInteger style )  
ProgressIndicatorSizeToFit( NSInteger tag )  
ProgressIndicatorIsDisplayWhenStopped( NSInteger tag ) = Boolean  
ProgressIndicatorSetDisplayWhenStopped( NSInteger tag, Boolean flag )
```





### See Also

[Control](#), [View](#)

### Apple documentation

[NSProgressIndicator](#)



Progress Indicator Styles		Regular		Small	
		w	h	w	h
	Determinate Bar	-	20	-	-
	Indeterminate Bar	-	20		-
	Determinate Circular	32	32	16	16
	Indeterminate Circular	32	32	16	16



### PropertyListSerialization

#### Functions

Serializing

```
PropertyListSerializationDataWithPropertyList( CTypeRef plistObj, NSPropertyListFormat format, ErrorRef *err ) = CFDataRef
```

Deserializing

```
PropertyListSerializationPropertyListWithData( CFDataRef dta, NSPropertyListReadOptions options, NSPropertyListFormat *format, ErrorRef *err ) = CTypeRef format and err params can be NULL
```

Validating

```
PropertyListSerializationPropertyListIsValidForFormat( CTypeRef plistObj, NSPropertyListFormat format ) = Boolean
```

#### Apple documentation

[NSPropertyListSerialization](#)



---

## Range

### Functions

Create a new CFRange from the specified values.

```
RangeMake( NSUInteger location, NSUInteger length ) = CFRange
```

Return the sum of the location and length of the range.

```
RangeMax( CFRange range ) = NSUInteger
```

Return a Boolean value that indicates whether a specified position is in a given range.

```
RangeLocation( CFRange range, NSUInteger location ) = Boolean
```

Return a Boolean value that indicates whether two given ranges are equal.

```
RangeEqual( CFRange range1, CFRange range2 ) = Boolean
```

Return the union of the specified ranges.

```
RangeUnion( CFRange range1, CFRange range2 ) = CFRange
```

Return the intersection of the specified ranges.

```
RangeIntersection( CFRange range1, CFRange range2 ) = CFRange
```

Return a range from a textual representation.

```
RangeFromString( CFStringRef string ) = CFRange
```

Return a string representation of a range.

```
RangeToString( CFRange range ) = CFStringRef
```

### Apple documentation

[NSRange](#)





---

## RegularExpression

### Functions

Create

```
RegularExpressionWithPattern( CFStringRef pattern, NSRegularExpressionOptions options, ErrorRef *err ) =  
RegularExpressionRef
```

Getting the regex and options

```
RegularExpressionPattern( RegularExpressionRef ref ) = CFStringRef  
RegularExpressionOptions( RegularExpressionRef ref ) = NSRegularExpressionOptions  
RegularExpressionNumberOfCaptureGroups( RegularExpressionRef ref ) = NSUInteger
```

Searching

```
RegularExpressionNumberOfMatches( RegularExpressionRef ref, CFStringRef string, NSMatchingOptions options, CFRange  
range ) = NSUInteger  
RegularExpressionEnumerateMatches( RegularExpressionRef ref, CFStringRef string, NSMatchingOptions options, CFRange  
range, ptr callback, ptr userData )  
RegularExpressionMatches( RegularExpressionRef ref, CFStringRef string, NSMatchingOptions options, CFRange range ) =  
CFArrayRef  
RegularExpressionFirstMatch( RegularExpressionRef ref, CFStringRef string, NSMatchingOptions options, CFRange  
range ) = TextCheckingResultRef  
RegularExpressionRangeOfFirstMatch( RegularExpressionRef ref, CFStringRef string, NSMatchingOptions options, CFRange  
range ) = CFRange
```

Replacing

```
RegularExpressionReplaceMatches( RegularExpressionRef ref, CFMutableStringRef string, NSMatchingOptions options,  
CFRange range, CFStringRef templ ) = NSUInteger  
RegularExpressionStringByReplacingMatches( RegularExpressionRef ref, CFStringRef string, NSMatchingOptions options,  
CFRange range, CFStringRef templ ) = CFStringRef
```

Escaping

```
RegularExpressionEscapedTemplate( CFStringRef string ) = CFStringRef  
RegularExpressionEscapedPattern( CFStringRef string ) = CFStringRef
```

Custom replace

```
RegularExpressionReplacementStringForResult( RegularExpressionRef ref, TextCheckingResultRef result, CFStringRef  
string, NSInteger offset, CFStringRef templ ) = CFStringRef
```

### Apple documentation

[NSRegularExpression](#)



---

## Responder

### Functions

Getting the undo manager

```
ResponderUndoManager( ResponderRef ref ) = UndoManagerRef
```

### Apple documentation

[NSResponder](#)



---

## RotationGestureRecognizer

### Functions

Init

```
RotationGestureRecognizerInit( ptr callback, ptr userData ) = RotationGestureRecognizerRef// autoreleased
```

Interpreting

```
RotationGestureRecognizerRotation( RotationGestureRecognizerRef ref ) = CGFloat
```

```
RotationGestureRecognizerSetRotation( RotationGestureRecognizerRef ref, CGFloat rotation )
```

```
RotationGestureRecognizerRotationInDegrees( RotationGestureRecognizerRef ref ) = CGFloat
```

```
RotationGestureRecognizerSetRotationDegrees( RotationGestureRecognizerRef ref, CGFloat rotation )
```

### Apple documentation

[NSRotationGestureRecognizer](#)



---

## RulerMarker

### Functions

Create

```
RulerMarkerWithRulerView( RulerViewRef ruler, CGFloat location, ImageRef image, CGPoint imageOrigin ) =  
RulerMarkerRef
```

Ruler view

```
RulerMarkerRuler( RulerMarkerRef ref ) = RulerViewRef
```

Image

```
RulerMarkerImage( RulerMarkerRef ref ) = ImageRef  
RulerMarkerSetImage( RulerMarkerRef ref, ImageRef image )  
RulerMarkerImageOrigin( RulerMarkerRef ref ) = CGPoint  
RulerMarkerSetImageOrigin( RulerMarkerRef ref, CGPoint origin )  
RulerMarkerImageRectInRuler( RulerMarkerRef ref ) = CGRect  
RulerMarkerThicknessRequiredInRuler( RulerMarkerRef ref ) = CGFloat
```

Movability

```
RulerMarkerIsMovable( RulerMarkerRef ref ) = Boolean  
RulerMarkerSetMovable( RulerMarkerRef ref, Boolean flag )  
RulerMarkerIsRemovable( RulerMarkerRef ref ) = Boolean  
RulerMarkerSetRemovable( RulerMarkerRef ref, Boolean flag )
```

Location

```
RulerMarkerLocation( RulerMarkerRef ref ) = CGFloat  
RulerMarkerSetLocation( RulerMarkerRef ref, CGFloat location )
```

Represented object

```
RulerMarkerRepresentedObject( RulerMarkerRef ref ) = ObjectRef  
RulerMarkerSetRepresentedObject( RulerMarkerRef ref, ObjectRef obj )
```

Drawing and event handling

```
RulerMarkerDrawRect( RulerMarkerRef ref, CGRect r )  
RulerMarkerIsDragging( RulerMarkerRef ref ) = Boolean
```

### Apple documentation

[NSRulerMarker](#)



## RulerView

### Functions

Create

```
RulerViewWithScrollView( NSInteger scrollTag, NSRulerOrientation orientation ) = RulerViewRef
```

Measurement units

```
RulerViewRegisterUnit( CFStringRef name, CFStringRef abbreviation, CGFloat unitToPointsFactor, CFArrayRef  
stepUpCycle, CFArrayRef stepDownCycle )  
RulerViewMeasurementUnits( RulerViewRef ref ) = CFStringRef  
RulerViewSetMeasurementUnits( RulerViewRef ref, CFStringRef unitName )
```

Client view

```
RulerViewClientView( RulerViewRef ref ) = ViewRef  
RulerViewSetClientView( RulerViewRef ref, ViewRef clientView )
```

Accessory view

```
RulerViewAccessoryView( RulerViewRef ref ) = ViewRef  
RulerViewSetAccessoryView( RulerViewRef ref, ViewRef accessoryView )
```

Zero mark position

```
RulerViewOriginOffset( RulerViewRef ref ) = CGFloat  
RulerViewSetOriginOffset( RulerViewRef ref, CGFloat offset )
```

Markers

```
RulerViewMarkers( RulerViewRef ref ) = CFArrayRef  
RulerViewSetMarkers( RulerViewRef ref, CFArrayRef markers )  
RulerViewAddMarker( RulerViewRef ref, RulerMarkerRef marker )  
RulerViewRemoveMarker( RulerViewRef ref, RulerMarkerRef marker )
```

Temporary ruler lines

```
RulerViewMoveRulerline( RulerViewRef ref, CGFloat fromLocation, CGFloat toLocation )
```

Drawing

```
RulerViewDrawHashMarksAndLabelsInRect( RulerViewRef ref, CGRect r )  
RulerViewDrawMarkersInRect( RulerViewRef ref, CGRect r )  
RulerViewInvalidateHashMarks( RulerViewRef ref )
```

Layout

```
RulerViewScrollView( RulerViewRef ref ) = NSInteger  
RulerViewSetScrollView( RulerViewRef ref, NSInteger scrollTag )  
RulerViewOrientation( RulerViewRef ref ) = NSRulerOrientation  
RulerViewSetOrientation( RulerViewRef ref, NSRulerOrientation orientation )  
RulerViewReservedThicknessForAccessoryView( RulerViewRef ref ) = CGFloat  
RulerViewSetReservedThicknessForAccessoryView( RulerViewRef ref, CGFloat thickness )  
RulerViewReservedThicknessForMarkers( RulerViewRef ref ) = CGFloat  
RulerViewSetReservedThicknessForMarkers( RulerViewRef ref, CGFloat thickness )  
RulerViewRuleThickness( RulerViewRef ref ) = CGFloat  
RulerViewSetRuleThickness( RulerViewRef ref, CGFloat thickness )  
RulerViewRequiredThickness( RulerViewRef ref ) = CGFloat  
RulerViewBaselineLocation( RulerViewRef ref ) = CGFloat  
RulerViewIsFlipped( RulerViewRef ref ) = Boolean
```

### Apple documentation

[NSRulerView](#)



---

## RunningApplication

### Functions

Getting instances

```
RunningApplicationWithIdentifier( pid_t pid ) = RunningApplicationRef
RunningApplicationsWithBundleIdentifier( CFStringRef identifier ) = CFArrayRef
RunningApplicationCurrentApplication = RunningApplicationRef
```

Activating

```
RunningApplicationIsActive( RunningApplicationRef ra ) = Boolean
RunningApplicationActivate( RunningApplicationRef ra, NSApplicationActivationOptions options ) = Boolean
RunningApplicationActivationPolicy( RunningApplicationRef ra ) = NSApplicationActivationPolicy
```

Hiding and unhiding

```
RunningApplicationHide( RunningApplicationRef ra ) = Boolean
RunningApplicationUnhide( RunningApplicationRef ra ) = Boolean
RunningApplicationIsHidden( RunningApplicationRef ra ) = Boolean
```

Info

```
RunningApplicationLocalizedName( RunningApplicationRef ra ) = CFStringRef
RunningApplicationIcon( RunningApplicationRef ra ) = ImageRef
RunningApplicationBundleIdentifier( RunningApplicationRef ra ) = CFStringRef
RunningApplicationBundleURL( RunningApplicationRef ra ) = CFURLRef
RunningApplicationExecutableArchitecture( RunningApplicationRef ra ) = NSInteger
RunningApplicationExecutableURL( RunningApplicationRef ra ) = CFURLRef
RunningApplicationLaunchDate( RunningApplicationRef ra ) = CFDateRef
RunningApplicationIsFinishedLaunching( RunningApplicationRef ra ) = Boolean
RunningApplicationProcessIdentifier( RunningApplicationRef ra ) = pid_t
RunningApplicationOwnsMenuBar( RunningApplicationRef ra ) = Boolean
```

Terminating

```
RunningApplicationForceTerminate( RunningApplicationRef ra ) = Boolean
RunningApplicationTerminate( RunningApplicationRef ra ) = Boolean
RunningApplicationIsTerminated( RunningApplicationRef ra ) = Boolean
RunningApplicationTerminateAutomaticallyTerminableApplications
```

### Apple documentation

[NSRunningApplication](#)



SavePanel

function/statement

Syntax

```
[1]
url = savepanel tag, msg, fileTypes, nameFldString, prompt, dirURL, sheetFlag

[2]
savepanel tag, msg, fileTypes, nameFldString, prompt, dirURL, sheetFlag
```

Description

Use this statement to:

- Build and run a new savepanel;
- Run an existing savepanel;
- Alter an existing savepanel’s characteristics.

Parameters

<i>tag</i>	Unique number to identify the savepanel. A negative value builds the panel invisibly and doesn’t run it until the statement is executed with a positive value. Note: Once the panel has been dismissed, its tag value is no longer valid.
<i>msg</i>	The message text.
<i>fileTypes</i>	The allowed file types. A semicolon-delimited string or array of types.
<i>nameFldString</i>	The user-editable filename currently shown in the name field.
<i>prompt</i>	The prompt of the default button.
<i>dirURL</i>	The directory shown in the panel.
<i>sheetFlag</i>	If this param is <code>_true</code> , the panel will be a sheet attached to the current output window. Note: A sheet does not return a response value from the savepanel statement, and instead posts a <code>_savePanelDidEnd</code> event to the user on dialog function.

Return value (non-sheet only)

A file URL.

Dialog Events

Event	Description
<code>_savePanelDidEnd</code>	This event is triggered when a savepanel sheet is dismissed.

Functions

```
SavePanelWithTag( NSInteger tag ) = SavePanelRef
SavePanelExists( NSInteger tag ) = Boolean

Init
SavePanelInit( NSInteger tag ) = SavePanelRef// autoreleased

Configure
SavePanelAccessoryView( NSInteger tag ) = ViewRef
SavePanelSetAccessoryView( NSInteger tag, ViewRef ref )
SavePanelTitle( NSInteger tag ) = CFStringRef
SavePanelSetTitle( NSInteger tag, CFStringRef title )
SavePanelPrompt( NSInteger tag ) = CFStringRef
SavePanelSetPrompt( NSInteger tag, CFStringRef prompt )
SavePanelNameFieldLabel( NSInteger tag ) = CFStringRef
SavePanelSetNameFieldLabel( NSInteger tag, CFStringRef label )
SavePanelMessage( NSInteger tag ) = CFStringRef
SavePanelSetMessage( NSInteger tag, CFStringRef message )
SavePanelCanCreateDirectories( NSInteger tag ) = Boolean
SavePanelSetCanCreateDirectories( NSInteger tag, Boolean flag )
SavePanelShowsHiddenFiles( NSInteger tag ) = Boolean
SavePanelSetShowsHiddenFiles( NSInteger tag, Boolean flag )
SavePanelShowsTagField( NSInteger tag ) = Boolean// macOS 10.9+
SavePanelSetShowsTagField( NSInteger tag, Boolean flag )// macOS 10.9+
```

```
SavePanelTagNames( NSInteger tag ) = CFArrayRef// macOS 10.9+
SavePanelSetTagNames( NSInteger tag, CFArrayRef tagNames )// macOS 10.9+
```

#### Panel content

```
SavePanelIsExtensionHidden( NSInteger tag ) = Boolean
SavePanelSetExtensionHidden( NSInteger tag, Boolean flag )
SavePanelDirectoryURL( NSInteger tag ) = CFURLRef
SavePanelSetDirectoryURL( NSInteger tag, CFURLRef url )
SavePanelCanSelectHiddenExtension( NSInteger tag ) = Boolean
SavePanelSetCanSelectHiddenExtension( NSInteger tag, Boolean flag )
SavePanelAllowedFileTypes( NSInteger tag ) = CFArrayRef
SavePanelSetAllowedFileTypes( NSInteger tag, CFArrayRef fileTypes )
SavePanelAllowsOtherFileTypes( NSInteger tag ) = Boolean
SavePanelSetAllowsOtherFileTypes( NSInteger tag, Boolean flag )
SavePanelTreatsFilePackagesAsDirectories( NSInteger tag ) = Boolean
SavePanelSetTreatsFilePackagesAsDirectories( NSInteger tag, Boolean flag )
```

#### Running

```
SavePanelValidateVisibleColumns( NSInteger tag )
```

#### User selection

```
SavePanelURL( NSInteger tag ) = CFURLRef
SavePanelIsExpanded( NSInteger tag ) = Boolean
SavePanelNameFieldStringValue( NSInteger tag ) = CFStringRef
SavePanelSetNameFieldStringValue( NSInteger tag, CFStringRef string )
```

#### Actions

```
SavePanelOK( NSInteger tag )
SavePanelCancel( NSInteger tag )
```

#### Custom

```
SavePanelRemove( NSInteger tag )
```

## Apple documentation

[NSSavePanel](#)





---

## Scanner

### Functions

Create

```
ScannerWithString( CFStringRef string ) = ScannerRef  
ScannerLocalizedWithString( CFStringRef string ) = ScannerRef
```

Get a scanner's string

```
ScannerString( ScannerRef ref ) = CFStringRef
```

Configure

```
ScannerLocation( ScannerRef ref ) = NSUInteger  
ScannerSetLocation( ScannerRef ref, NSUInteger location )  
ScannerCaseSensitive( ScannerRef ref ) = Boolean  
ScannerSetCaseSensitive( ScannerRef ref, Boolean flag )  
ScannerCharactersToBeSkipped( ScannerRef ref ) = CFCharacterSetRef  
ScannerSetCharactersToBeSkipped( ScannerRef ref, CFCharacterSetRef set )  
ScannerLocale( ScannerRef ref ) = CFLocaleRef  
ScannerSetLocale( ScannerRef ref, CFLocaleRef locale )
```

Scanning a string

```
ScannerScanCharactersFromSet( ScannerRef ref, CFCharacterSetRef set, CFStringRef *result ) = Boolean// result may be NULL  
ScannerScanUpToCharactersFromSet( ScannerRef ref, CFCharacterSetRef set, CFStringRef *result ) = Boolean// result may be NULL  
ScannerScanDecimal( ScannerRef ref, NSDecimal *result ) = Boolean// result may be NULL  
ScannerScanDouble( ScannerRef ref, double *result ) = Boolean// result may be NULL  
ScannerScanFloat( ScannerRef ref, float *result ) = Boolean// result may be NULL  
ScannerScanHexDouble( ScannerRef ref, double *result ) = Boolean// result may be NULL  
ScannerScanHexFloat( ScannerRef ref, float *result ) = Boolean// result may be NULL  
ScannerScanHexInt( ScannerRef ref, unsigned long *result ) = Boolean// result may be NULL  
ScannerScanHexLongLong( ScannerRef ref, UInt64 *result ) = Boolean// result may be NULL  
ScannerScanInteger( ScannerRef ref, NSInteger *result ) = Boolean// result may be NULL  
ScannerScanLongLong( ScannerRef ref, UInt64 *result ) = Boolean// result may be NULL  
ScannerScanString( ScannerRef ref, CFStringRef string, CFStringRef *result ) = Boolean// result may be NULL  
ScannerScanUpToString( ScannerRef ref, CFStringRef string, CFStringRef *result ) = Boolean// result may be NULL  
ScannerIsAtEnd( ScannerRef ref ) = Boolean
```

### Apple documentation

[NSScanner](#)



---

## Screen

### Functions

Screen objects

```
ScreenMainScreen = ScreenRef
ScreenDeepestScreen = ScreenRef
ScreenScreens = CFArrayRef
```

Info

```
ScreenDepth( ScreenRef scrn ) = NSInteger
ScreenFrame( ScreenRef scrn ) = CGRect
ScreenDeviceDescription( ScreenRef scrn ) = CFDictionaryRef
ScreenVisibleFrame( ScreenRef scrn ) = CGRect
ScreenCanRepresentDisplayGamut( ScreenRef scrn, NSInteger gamut ) = Boolean// macOS 10.12+
ScreenScreensHaveSeparateSpaces = Boolean// macOS 10.9+
```

Backing coordinate conversion

```
ScreenBackingAlignedRect( ScreenRef scrn, CGRect rect, NSAlignmentOptions options ) = CGRect
ScreenBackingScaleFactor( ScreenRef scrn ) = CGFloat
ScreenConvertRectFromBacking( ScreenRef scrn, CGRect rect ) = CGRect
ScreenConvertRectToBacking( ScreenRef scrn, CGRect rect ) = CGRect
```

Extended dynamic range

```
ScreenMaximumExtendedDynamicRangeColorComponentValue( ScreenRef scrn ) = CGFloat// macOS 10.11+
```

Convenience

```
ScreenMainFrame = CGRect
```

### Apple documentation

[NSScreen](#)



---

## Scroller

### Functions

Scroller size

```
ScrollerWidth( NSControlSize size, NSScrollerStyle style ) = CGFloat  
ScrollerControlSize( ScrollerRef ref ) = NSControlSize
```

Knob position

```
ScrollerKnobProportion( ScrollerRef ref ) = CGFloat  
ScrollerSetKnobProportion( ScrollerRef ref, CGFloat proportion )
```

Calculate layout

```
ScrollerRectForPart( ScrollerRef ref, NSScrollerPart part ) = CGRect  
ScrollerTestPart( ScrollerRef ref, CGPoint pt ) = NSScrollerPart  
ScrollerUsableParts( ScrollerRef ref ) = NSUsableScrollerParts
```

Draw scroller parts

```
ScrollerDrawKnobSlotInRect( ScrollerRef ref, CGRect r, Boolean highlight )
```

Event handling

```
ScrollerHitPart( ScrollerRef ref ) = NSScrollerPart
```

Presentation style

```
ScrollerPreferredScrollerStyle = NSScrollerStyle  
ScrollerScrollerStyle( ScrollerRef ref ) = NSScrollerStyle  
ScrollerKnobStyle( ScrollerRef ref ) = NSScrollerKnobStyle
```

Type properties

```
ScrollerIsCompatibleWithOverlayScrollers = Boolean
```

### Apple documentation

[NSScroller](#)



## ScrollView

statement

### Syntax

**scrollview** *tag*, *rect*, *borderType*, *documentTag*, *wndTag*

### Description

The **scrollview** statement puts a new scrollview in the current output cocoa window, or alters an existing scrollview's characteristics.

### Parameters

<i>tag</i>	A number to identify the scrollview. A negative tag hides the scrollview.
<i>rect</i>	Origin and size of the scrollview in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>borderType</i>	The scrollview's border type. <a href="#">NSNoBorder</a> (default) <a href="#">NSLineBorder</a> <a href="#">NSBezelBorder</a> <a href="#">NSGrooveBorder</a>
<i>documentTag</i>	The scrollview's document.
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Functions

```
ScrollViewWithTag( NSInteger tag ) = ScrollViewRef
ScrollViewExists( NSInteger tag ) = Boolean
```

#### Init

```
ScrollViewInit( NSInteger tag, CGRect r ) = ScrollViewRef// autoreleased
```

#### Component size

```
ScrollViewContentSize( NSInteger tag ) = CGSize
ScrollViewDocumentVisibleRect( NSInteger tag ) = CGRect
```

#### Graphics attributes

```
ScrollViewBackgroundColor( NSInteger tag ) = ColorRef
ScrollViewSetBackgroundColor( NSInteger tag, ColorRef col )
ScrollViewDrawsBackground( NSInteger tag ) = Boolean
ScrollViewSetDrawsBackground( NSInteger tag, Boolean flag )
ScrollViewBorderType( NSInteger tag ) = NSBorderType
ScrollViewSetBorderType( NSInteger tag, NSBorderType type )
ScrollViewDocumentCursor( NSInteger tag ) = CursorRef
ScrollViewSetDocumentCursor( NSInteger tag, CursorRef ref )
```

#### Views

```
ScrollViewContentView( NSInteger tag ) = ClipViewRef
ScrollViewSetContentView( NSInteger tag, ClipViewRef clip )
ScrollViewSetDocumentView( NSInteger scrollTag, NSInteger docTag )
ScrollViewAddFloatingSubview( NSInteger scrollTag, NSInteger subviewTag, NSEventGestureAxis axis )macOS 10.9+
```

#### Scrollers

```
ScrollViewHorizontalScroller( NSInteger tag ) = ScrollerRef
ScrollViewHasHorizontalScroller( NSInteger tag ) = Boolean
ScrollViewSetHasHorizontalScroller( NSInteger tag, Boolean flag )
ScrollViewVerticalScroller( NSInteger tag ) = ScrollerRef
ScrollViewHasVerticalScroller( NSInteger tag ) = Boolean
ScrollViewSetHasVerticalScroller( NSInteger tag, Boolean flag )
ScrollViewAutohidesScrollers( NSInteger tag ) = Boolean
ScrollViewSetAutohidesScrollers( NSInteger tag, Boolean flag )
```

#### Rulers

```
ScrollViewHasHorizontalRuler( NSInteger tag ) = Boolean
ScrollViewSetHasHorizontalRuler( NSInteger tag, Boolean flag )
ScrollViewHorizontalRulerView( NSInteger tag ) = RulerViewRef
ScrollViewSetHorizontalRulerView( NSInteger tag, RulerViewRef ruler )
```

```
UIScrollViewHasVerticalRuler( NSInteger tag ) = Boolean
UIScrollViewSetHasVerticalRuler( NSInteger tag, Boolean flag )
UIScrollViewVerticalRulerView( NSInteger tag ) = RulerViewRef
UIScrollViewSetVerticalRulerView( NSInteger tag, RulerViewRef ruler )
UIScrollViewRulersVisible( NSInteger tag ) = Boolean
UIScrollViewSetRulersVisible( NSInteger tag, Boolean flag )
```

Insets

```
UIScrollViewAutomaticallyAdjustsContentInsets( NSInteger tag ) = Boolean// macOS 10.10+
UIScrollViewSetAutomaticallyAdjustsContentInsets( NSInteger tag, Boolean flag )// macOS 10.10+
UIScrollViewContentInsets( NSInteger tag ) = NSEdgeInsets// macOS 10.10+
UIScrollViewSetContentInsets( NSInteger tag, NSEdgeInsets insets )// macOS 10.10+
UIScrollViewScrollerInsets( NSInteger tag ) = NSEdgeInsets// macOS 10.10+
UIScrollViewSetScrollerInsets( NSInteger tag, NSEdgeInsets insets )// macOS 10.10+
```

Style

```
UIScrollViewScrollerKnobStyle( NSInteger tag ) = NSScrollerKnobStyle
UIScrollViewSetScrollerKnobStyle( NSInteger tag, NSScrollerKnobStyle style )
UIScrollViewScrollerStyle( NSInteger tag ) = NSScrollerStyle
UIScrollViewSetScrollerStyle( NSInteger tag, NSScrollerStyle style )
```

Scrolling behavior

```
UIScrollViewLineScroll( NSInteger tag ) = CGFloat
UIScrollViewSetLineScroll( NSInteger tag, CGFloat value )
UIScrollViewHorizontalLineScroll( NSInteger tag ) = CGFloat
UIScrollViewSetHorizontalLineScroll( NSInteger tag, CGFloat value )
UIScrollViewVerticalLineScroll( NSInteger tag ) = CGFloat
UIScrollViewSetVerticalLineScroll( NSInteger tag, CGFloat value )
UIScrollViewPageScroll( NSInteger tag ) = CGFloat
UIScrollViewSetPageScroll( NSInteger tag, CGFloat value )
UIScrollViewHorizontalPageScroll( NSInteger tag ) = CGFloat
UIScrollViewSetHorizontalPageScroll( NSInteger tag, CGFloat value )
UIScrollViewVerticalPageScroll( NSInteger tag ) = CGFloat
UIScrollViewSetVerticalPageScroll( NSInteger tag, CGFloat value )
UIScrollViewScrollsDynamically( NSInteger tag ) = Boolean
UIScrollViewSetScrollsDynamically( NSInteger tag, Boolean flag )
UIScrollViewScrollWheel( NSInteger tag, CocoaEventRef evnt )
```

Updating display

```
UIScrollViewReflectScrolledClipView( NSInteger tag )
```

Find bar

```
UIScrollViewFindBarPosition( NSInteger tag ) = NSScrollViewFindBarPosition
UIScrollViewSetFindBarPosition( NSInteger tag, NSScrollViewFindBarPosition position )
```

Predominant behavior

```
UIScrollViewUsesPredominantAxisScrolling( NSInteger tag ) = Boolean
UIScrollViewSetUsesPredominantAxisScrolling( NSInteger tag, Boolean flag )
```

Elasticity

```
UIScrollViewHorizontalScrollElasticity( NSInteger tag ) = NSInteger
UIScrollViewSetHorizontalScrollElasticity( NSInteger tag, NSInteger elasticity )
UIScrollViewVerticalScrollElasticity( NSInteger tag ) = NSInteger
UIScrollViewSetVerticalScrollElasticity( NSInteger tag, NSInteger elasticity )
```

Flash scrollers

```
UIScrollViewFlashScrollers( NSInteger tag )
```

Zooming

```
UIScrollViewAllowsMagnification( NSInteger tag ) = BooleanmacOS 10.8+
UIScrollViewSetAllowsMagnification( NSInteger tag, Boolean flag )macOS 10.8+
UIScrollViewMagnification( NSInteger tag ) = CGFloatmacOS 10.8+
UIScrollViewSetMagnification( NSInteger tag, CGFloat value )macOS 10.8+
UIScrollViewMagnifyToFitRect( NSInteger tag, CGRect r )macOS 10.8+
UIScrollViewMaxMagnification( NSInteger tag ) = CGFloatmacOS 10.8+
UIScrollViewSetMaxMagnification( NSInteger tag, CGFloat value )macOS 10.8+
UIScrollViewMinMagnification( NSInteger tag ) = CGFloatmacOS 10.8+
UIScrollViewSetMinMagnification( NSInteger tag, CGFloat value )macOS 10.8+
UIScrollViewSetMagnificationCenteredAtPoint( NSInteger tag, CGFloat value, CGPoint pt )macOS 10.8+
```

Convenience

```
UIScrollViewScrollToPoint( NSInteger tag, CGPoint pt )
```

## See Also

[View](#)

## Apple documentation

[NSScrollView](#)



## SearchField

statement

### Syntax

```
searchfield tag, enabled, text, rect, wndTag
```

### Description

The **searchfield** statement puts a new searchfield in the current output cocoa window, or alters an existing searchfield's characteristics.

### Parameters

<i>tag</i>	A number to identify the searchfield. A negative tag hides the searchfield.
<i>enabled</i>	Enables or disables the searchfield.
<i>text</i>	The searchfield's text.
<i>rect</i>	Origin and size of the searchfield in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

Event	Description
<code>_textFieldXxxx</code>	See <a href="#">textfield</a> .
<code>_searchFieldDidStartSearching</code>	
<code>_searchFieldDidEndSearching</code>	

### Functions

```
SearchFieldWithTag( NSInteger tag ) = SearchFieldRef  
SearchFieldExists( NSInteger tag ) = Boolean
```

Init

```
SearchFieldInit( NSInteger tag, CGRect r ) = SearchFieldRef// autoreleased
```

Menu template

```
SearchFieldSetSearchMenuTemplate( NSInteger tag, NSInteger menuIndex )macOS 10.10
```

Modes

```
SearchFieldSendsSearchStringImmediately( NSInteger tag ) = Boolean// macOS 10.10+  
SearchFieldSetSendsSearchStringImmediately( NSInteger tag, Boolean flag )macOS 10.10  
SearchFieldSendsWholeSearchString( NSInteger tag ) = Boolean// macOS 10.10+  
SearchFieldSetSendsWholeSearchString( NSInteger tag, Boolean flag )macOS 10.10
```

Recent searches

```
SearchFieldRecentSearches( NSInteger tag ) = CFArrayRef
```

Recent search strings

```
SearchFieldMaximumRecents( NSInteger tag ) = NSInteger// macOS 10.10+  
SearchFieldSetMaximumRecents( NSInteger tag, NSInteger value )macOS 10.10  
SearchFieldRecentsAutosaveName( NSInteger tag ) = CFStringRef  
SearchFieldSetRecentsAutosaveName( NSInteger tag, CFStringRef name )
```

Instance properties

```
SearchFieldCentersPlaceholder( NSInteger tag ) = Boolean// macOS 10.11+  
SearchFieldSetCentersPlaceholder( NSInteger tag, Boolean flag )macOS 10.11
```

### See Also

[Control](#), [View](#), [TextField](#)

### Apple documentation

[NSSearchField](#)



SecureTextField

statement

Syntax

`securetextfield tag, enabled, text, rect, wndTag`

Description

The **securetextfield** statement puts a new securetextfield in the current output cocoa window, or alters an existing securetextfield's characteristics.

Parameters

<i>tag</i>	A number to identify the securetextfield. A negative tag hides the securetextfield.
<i>enabled</i>	Enables or disables the securetextfield.
<i>text</i>	The securetextfield's text.
<i>rect</i>	Origin and size of the securetextfield in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

Dialog Events

See [TextField](#)

Functions

`SecureTextFieldWithTag( NSInteger tag ) = SecureTextFieldRef`

`SecureTextFieldExists( NSInteger tag ) = Boolean`

Init

`SecureTextFieldInit( NSInteger tag, CGRect r ) = SecureTextFieldRef// autoreleased`

Convenience

`SecureTextFieldEchosBullets( NSInteger tag ) = Boolean`

`SecureTextFieldSetEchosBullets( NSInteger tag, Boolean flag )`

See [TextField](#)

See Also

[Control](#), [View](#)

Apple documentation

[NSSecureTextField](#)



## SegmentedControl

statement

### Syntax

```
segmentedcontrol tag, enabled, index, rect, segments, wndTag
```

### Description

The **segmentedcontrol** statement puts a new segmentedcontrol in the current output cocoa window, or alters an existing segmentedcontrol's characteristics.

### Parameters

<i>tag</i>	A number to identify the segmentedcontrol. A negative tag hides the segmentedcontrol.
<i>enabled</i>	Enable or disable the segmentedcontrol.
<i>index</i>	The index of the segmentedcontrol's currently selected segment.
<i>rect</i>	Origin and size of the segmentedcontrol in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>segments</i>	The number of segments in the control (default = 3).
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

[\\_btnClick](#)

### Functions

```
SegmentedControlWithTag( NSInteger tag ) = SegmentedControlRef  
SegmentedControlExists( NSInteger tag ) = Boolean
```

#### Init

```
SegmentedControlInit( NSInteger tag, CGRect r ) = SegmentedControlRef// autoreleased
```

#### Behavior

```
SegmentedControlTrackingMode( NSInteger tag ) = NSSegmentSwitchTracking// macOS 10.10.3  
SegmentedControlSetTrackingMode( NSInteger tag, NSSegmentSwitchTracking mode )  
SegmentedControlStyle( NSInteger tag ) = NSSegmentStyle  
SegmentedControlSetStyle( NSInteger tag, NSSegmentStyle style )
```

#### Number of segments

```
SegmentedControlSegmentCount( NSInteger tag ) = NSInteger  
SegmentedControlSetSegmentCount( NSInteger tag, NSInteger count )
```

#### Text

```
SegmentedControlLabelForSegment( NSInteger tag, NSInteger index ) = CFStringRef  
SegmentedControlSetSegmentLabel( NSInteger tag, NSInteger index, CFStringRef string )  
SegmentedControlSegmentAlignment( NSInteger tag, NSInteger index ) = NSTextAlignment// macOS 10.13+  
SegmentedControlSetSegmentAlignment( NSInteger tag, NSInteger index, NSTextAlignment value )// macOS 10.13+
```

#### Image

```
SegmentedControlImageForSegment( NSInteger tag, NSInteger index ) = ImageRef  
SegmentedControlSetSegmentImage( NSInteger tag, NSInteger index, ImageRef image )  
SegmentedControlImageScaling( NSInteger tag, NSInteger index ) = NSImageScaling  
SegmentedControlSetImageScaling( NSInteger tag, NSInteger index, NSImageScaling scaling )
```

#### Menu

```
SegmentedControlMenuForSegment( NSInteger tag, NSInteger index ) = CocoaMenuRef  
SegmentedControlSetSegmentMenu( NSInteger tag, NSInteger index, CocoaMenuRef menu )  
SegmentedControlSetSegmentMenuIndex( NSInteger tag, NSInteger index, NSInteger menuIndex )  
SegmentedControlSegmentShowsMenuIndicator( NSInteger tag, NSInteger index ) = Boolean// macOS 10.13+  
SegmentedControlSetSegmentShowsMenuIndicator( NSInteger tag, NSInteger index, Boolean flag )// macOS 10.13+  
SegmentedControlIsSpringLoaded( NSInteger tag ) = Boolean// macOS 10.10.3+  
SegmentedControlSetSpringLoaded( NSInteger tag, Boolean flag )// macOS 10.10.3+
```

#### Selected segment



```
SegmentedControlSelectedSegment( NSInteger tag ) = NSInteger
SegmentedControlIndexOfSelectedItem( NSInteger tag ) = NSInteger// macOS 10.13+
SegmentedControlSelectSegmentWithTag( NSInteger ctrlTag, NSInteger segTag ) = Boolean
SegmentedControlSetSegmentSelected( NSInteger tag, NSInteger index, Boolean flag )
SegmentedControlIsSegmentSelected( NSInteger tag, NSInteger index ) = Boolean
SegmentedControlSelectedSegmentBezelColor( NSInteger tag ) = ColorRef// macOS 10.12.1+
SegmentedControlSetSelectedSegmentBezelColor( NSInteger tag, ColorRef col )// macOS 10.12.1+
SegmentedControlSelectedSegmentDoubleValue( NSInteger tag, NSInteger index ) = double// macOS 10.10.3+
```

#### Spacing

```
SegmentedControlSegmentWidth( NSInteger tag, NSInteger index ) = CGFloat
SegmentedControlSetSegmentWidth( NSInteger tag, NSInteger index, CGFloat value )
SegmentedControlSetSegmentDistribution( NSInteger tag, NSSegmentDistribution value )// macOS 10.13+
```

#### Enable

```
SegmentedControlIsSegmentEnabled( NSInteger tag, NSInteger index ) = Boolean
SegmentedControlSetSegmentEnabled( NSInteger tag, NSInteger index, Boolean flag )
```

#### Tags and tooltips

```
SegmentedControlSegmentTag( NSInteger tag, NSInteger index ) = NSInteger// macOS 10.13+
SegmentedControlSetSegmentTag( NSInteger tag, NSInteger index, NSInteger segTag )// macOS 10.13+
SegmentedControlSegmentToolTip( NSInteger tag, NSInteger index ) = CFStringRef// macOS 10.13+
SegmentedControlSetSegmentToolTip( NSInteger tag, NSInteger index, CFStringRef string )// macOS 10.13+
```

#### Custom

```
SegmentedControlSetSegment( NSInteger tag, NSInteger index, NSInteger enabled, CFStringRef label, ImageRef image,
NSInteger imageScaling, NSInteger width )
```

## See Also

[Control](#), [View](#)

## Apple documentation

[NSSegmentedControl](#)



## Set

### Functions

Create

```
SetWithArray( CFArrayRef array ) = CFSetRef
SetWithObject( CTypeRef obj ) = CFSetRef
SetWithObjects( CTypeRef obj1, ... ) = CFSetRef
SetByAddingObject( CFSetRef set, CTypeRef obj ) = CFSetRef
SetByAddingObjectsFromSet( CFSetRef set1, CFSetRef set2 ) = CFSetRef
SetByAddingObjectsFromArray( CFSetRef set, CFArrayRef array ) = CFSetRef
```

Count

```
SetCount( CFSetRef set ) = NSInteger
```

Accessing members

```
SetAllObjects( CFSetRef set ) = CFArrayRef
SetAnyObject( CFSetRef set ) = CTypeRef
SetContainsObject( CFSetRef set, CTypeRef obj ) = Boolean
SetMember( CFSetRef set, CTypeRef obj ) = CTypeRef
SetObjectEnumerator( CFSetRef set ) = EnumeratorRef
```

Compare

```
SetIsSubsetOfSet( CFSetRef set1, CFSetRef set2 ) = Boolean
SetIntersectsSet( CFSetRef set1, CFSetRef set2 ) = Boolean
SetIsEqualToSet( CFSetRef set1, CFSetRef set2 ) = Boolean
SetValueForKey( CFSetRef set, CFStringRef key ) = CTypeRef
SetSetValueForKey( CFSetRef set, CTypeRef value, CFStringRef key )
```

Sorted array

```
SetSortedArrayUsingDescriptors( CFSetRef set, CFArrayRef sortDescriptors ) = CFArrayRef
```

Description

```
SetDescription( CFSetRef set ) = CFStringRef
```

• Mutable set •

Create

```
SetWithCapacity( NSInteger length ) = CFMutableSetRef
```

Add and remove

```
SetAddObject( CFMutableSetRef set, CTypeRef obj )
SetRemoveObject( CFMutableSetRef set, CTypeRef obj )
SetRemoveAllObjects( CFMutableSetRef set )
SetAddObjectsFromArray( CFMutableSetRef set, CFArrayRef array )
```

Combine/recombine

```
SetUnionSet( CFMutableSetRef set1, CFSetRef set2 )
SetMinusSet( CFMutableSetRef set1, CFSetRef set2 )
SetIntersectSet( CFMutableSetRef set1, CFSetRef set2 )
SetSet( CFMutableSetRef set1, CFSetRef set2 )
```

### Apple documentation

[NSSet](#)

[NSMutableSet](#)



---

## Shadow

### Functions

Create

```
ShadowInit = ShadowRef// autoreleased
```

Managing shadow

```
ShadowOffset( ShadowRef shadow ) = CGSize  
ShadowSetOffset( ShadowRef shadow, CGSize offset )  
ShadowBlurRadius( ShadowRef shadow ) = CGFloat  
ShadowSetBlurRadius( ShadowRef shadow, CGFloat radius )  
ShadowColor( ShadowRef shadow ) = ColorRef  
ShadowSetColor( ShadowRef shadow, ColorRef col )
```

Setting

```
ShadowSet( ShadowRef shadow )
```

Convenience

```
ShadowWithAttributes( CGSize offset, CGFloat blurRadius, ColorRef col ) = ShadowRef  
ShadowSetWithAttributes( CGSize offset, CGFloat blurRadius, ColorRef col )
```

### Apple documentation

[NSShadow](#)



## Slider

statement

### Syntax

**slider** *tag, enabled, value, rect, min, max, tickMarks, wndTag*

### Description

The **slider** statement puts a new slider in the current output cocoa window, or alters an existing slider's characteristics.

### Parameters

<i>tag</i>	A number to identify the slider. A negative tag hides the slider.
<i>enabled</i>	Enables or disables the slider.
<i>value</i>	The value that indicates the current extent of the slider.
<i>rect</i>	Origin and size of the slider in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>min</i>	The minimum value of the slider (default = 0).
<i>max</i>	The maximum value of the slider (default = 100).
<i>tickMarks</i>	The number of tickmarks to be displayed (default = 0).
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

[\\_btnClick](#)

### Functions

See [control](#)

```
SliderWithTag( NSInteger tag ) = SliderRef  
SliderExists( NSInteger tag ) = Booleanf
```

```
Init  
SliderInit( NSInteger tag, CGRect r ) = SliderRef// autoreleased
```

#### Appearance

```
SliderType( NSInteger tag ) = NSSliderType  
SliderSetType( NSInteger tag, NSSliderType type )  
SliderAltIncrementValue( NSInteger tag ) = double  
SliderSetAltIncrementValue( NSInteger tag, double value )  
SliderKnobThickness( NSInteger tag ) = CGFloat  
SliderIsVertical( NSInteger tag ) = Boolean
```

#### Limits

```
SliderMaxValue( NSInteger tag ) = double  
SliderSetMaxValue( NSInteger tag, double value )  
SliderMinValue( NSInteger tag ) = double  
SliderSetMinValue( NSInteger tag, double value )
```

#### Tick marks

```
SliderAllowsTickMarkValuesOnly( NSInteger tag ) = Boolean  
SliderSetAllowsTickMarkValuesOnly( NSInteger tag, Boolean flag ) default = _false  
SliderClosestTickMarkValueToValue( NSInteger tag, double value ) = double  
SliderIndexOfTickMarkAtPoint( NSInteger tag, CGPoint pt ) = NSInteger  
SliderNumberOfTickMarks( NSInteger tag ) = NSInteger  
SliderSetNumberOfTickMarks( NSInteger tag, NSInteger value )  
SliderRectOfTickMarkAtIndex( NSInteger tag, NSInteger index ) = CGRect  
SliderTickMarkPosition( NSInteger tag ) = NSTickMarkPosition  
SliderSetTickMarkPosition( NSInteger tag, NSTickMarkPosition position )  
SliderTickMarkValueAtIndex( NSInteger tag, NSInteger index ) = double
```

#### Instance properties

```
SliderTrackFillColor( NSInteger tag ) = ColorRef// macOS 10.12.2+
```

```
SliderSetTrackFillColor( NSInteger tag, ColorRef col )// macOS 10.12.2+
```

## See Also

[Control](#), [View](#)

## Apple documentation

[NSSlider](#)



---

## SortDescriptor

### Functions

Init

```
SortDescriptorWithKey( CFStringRef key, Boolean ascending ) = SortDescriptorRef
```

```
SortDescriptorWithKeyAndSelector( CFStringRef key, Boolean ascending, CFStringRef selector ) = SortDescriptorRef
```

Info

```
SortDescriptorAscending( SortDescriptorRef sd ) = Boolean
```

```
SortDescriptorKey( SortDescriptorRef sd ) = CFStringRef
```

Using

```
SortDescriptorCompare( SortDescriptorRef sd, CTypeRef obj1, CTypeRef obj2 ) = NSComparisonResult
```

```
SortDescriptorReversedSortDescriptor( SortDescriptorRef sd ) = SortDescriptorRef
```

```
SortDescriptorAllowEvaluation( SortDescriptorRef sd )
```

### Apple documentation

[NSSortDescriptor](#)



---

## Sound

### Dialog Events

[\\_soundDidFinishPlaying](#)

### Functions

#### Create

```
SoundCanInitWithPasteboard( CocoaPasteboardRef pb ) = Boolean
SoundWithContentsOfURL( CFURLRef url, Boolean byReference ) = SoundRef
SoundWithData( CFDataRef data ) = SoundRef
SoundWithPasteboard( CocoaPasteboardRef pb ) = SoundRef
```

#### Configure

```
SoundName( SoundRef snd ) = CFStringRef
SoundSetName( SoundRef snd, CFStringRef name ) = Boolean
SoundVolume( SoundRef snd ) = float
SoundSetVolume( SoundRef snd, float value )
SoundCurrentTime( SoundRef snd ) = CFTimeInterval
SoundSetCurrentTime( SoundRef snd, CFTimeInterval ti )
SoundLoops( SoundRef snd ) = Boolean
SoundSetLoops( SoundRef snd, Boolean flag )
SoundPlaybackDeviceIdentifier( SoundRef snd ) = CFStringRef
SoundSetPlaybackDeviceIdentifier( SoundRef snd, CFStringRef identifier )
```

#### Info

```
SoundUnfilteredTypes = CFArrayRef
SoundNamed( CFStringRef name ) = SoundRef
SoundDuration( SoundRef snd ) = CFTimeInterval
```

#### Play

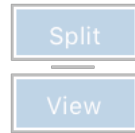
```
SoundIsPlaying( SoundRef snd ) = Boolean
SoundPause( SoundRef snd ) = Boolean
SoundPlay( SoundRef snd ) = Boolean
SoundResume( SoundRef snd ) = Boolean
SoundStop( SoundRef snd ) = Boolean
```

#### Write

```
SoundWriteToPasteboard( SoundRef snd, CocoaPasteboardRef pb )
```

### Apple documentation

[NSSound](#)



## SplitView

statement

### Syntax

**splitview** *tag, rect, dividerStyle, vertical, wndTag*

### Description

The **splitview** statement puts a new splitview in the current output cocoa window, or alters an existing splitview's characteristics.

### Parameters

<i>tag</i>	A number to identify the splitview. A negative tag hides the splitview.
<i>rect</i>	Origin and size of the splitview in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) CGRect value
<i>dividerStyle</i>	The splitview's divider style. NSSplitViewDividerStyleThick NSSplitViewDividerStyleThin NSSplitViewDividerStylePaneSplitter (default)
<i>vertical</i>	A boolean to indicate whether the splitview should be split vertically (default = _false)
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

Event	Description
_splitViewResizeSubviewsWithOldSize	Allows the delegate to specify custom sizing behavior for the subviews of the NSSplitView sender.
_splitViewWillResizeSubviews	Invoked by the default notification center to notify the delegate that the splitview will resize its subviews.
_splitViewDidResizeSubviews	Invoked by the default notification center to notify the delegate that the splitview did resize its subviews.
_splitViewCanCollapseSubview	Allows the delegate to determine whether the user can collapse and uncollapse subview. The subview tag is retrieved with DialogEventViewTag. Call DialogEventSetBool(_true) if the subview should collapse when the user drags a divider beyond the halfway mark between its min size and its edge.
_splitViewShouldCollapseSubview	Invoked to allow a delegate to determine if a subview should collapse in response to a double click.
_splitViewShouldAdjustSizeOfSubview	Allows the delegate to specify whether the subview should be resized.
_splitViewEffectiveRect	Allows the delegate to modify the rectangle in which mouse clicks initiate divider dragging.
_splitViewShouldHideDividerAtIndex	Allows the delegate to determine whether a divider can be dragged or adjusted off the edge of the split view.
_splitViewAdditionalEffectiveRectOfDividerAtIndex	Allows the delegate to return an additional rectangle in which mouse clicks will initiate divider dragging.
_splitViewConstrainMaxCoordinate	Allows the delegate for sender to constrain the maximum coordinate limit of a divider when the user drags it.
_splitViewConstrainMinCoordinate	Allows the delegate for sender to constrain the minimum coordinate limit of a divider when the user drags it.
_splitViewConstrainSplitPosition	Allows the delegate for sender to constrain the divider to certain positions.



## Functions

```
SplitViewWithTag( NSInteger tag ) = SplitViewRef  
SplitViewExists( NSInteger tag ) = Boolean
```

```
Init  
SplitViewInit( NSInteger tag, CGRect r ) = SplitViewRef// autoreleased
```

```
Subviews  
SplitViewAdjustSubviews( NSInteger tag )  
SplitViewIsPaneCollapsed( NSInteger tag, NSInteger paneIndex ) = Boolean  
SplitViewSetHoldingPriority( NSInteger tag, NSInteger paneIndex, float priority )
```

```
Divider orientation  
SplitViewIsVertical( NSInteger tag ) = Boolean  
SplitViewSetVertical( NSInteger tag, Boolean flag )
```

```
Configure and draw dividers  
SplitViewDividerStyle( NSInteger tag ) = NSSplitViewDividerStyle  
SplitViewSetDividerStyle( NSInteger tag, NSSplitViewDividerStyle style )  
SplitViewDividerThickness( NSInteger tag ) = CGFloat
```

```
Saving subview positions  
SplitViewAutosaveName( NSInteger tag ) = CFStringRef  
SplitViewSetAutosaveName( NSInteger tag, CFStringRef name )
```

```
Split position  
SplitViewMinPossiblePositionOfDivider( NSInteger tag, NSInteger index ) = CGFloat  
SplitViewMaxPossiblePositionOfDivider( NSInteger tag, NSInteger index ) = CGFloat  
SplitViewSetPositionOfDivider( NSInteger tag, NSInteger index, CGFloat position )
```

```
Convenience  
SplitViewAddSubview( NSInteger tag, NSInteger paneIndex, NSInteger subviewTag )
```

## See Also

[View](#)

## Apple documentation

[NSSplitView](#)



---

## StatusBar

### Functions

System-wide instance

```
StatusBarSystem = StatusBarRef
```

Status items

```
StatusBarStatusItem( StatusBarRef bar, CGFloat length ) = StatusItemRef
```

```
StatusBarRemoveStatusItem( StatusBarRef bar, StatusItemRef item )
```

Attributes

```
StatusBarIsVertical( StatusBarRef bar ) = Boolean
```

```
StatusBarThickness( StatusBarRef bar ) = CGFloat
```

### Apple documentation

[NSStatusBar](#)



---

### StatusBarButton

#### Functions

Instance properties

```
StatusBarButtonAppearsDisabled( StatusBarButtonRef ref ) = Boolean
```

```
StatusBarButtonSetAppearsDisabled( StatusBarButtonRef ref, Boolean flag )
```

#### Apple documentation

[NSStatusBarButton](#)



---

## StatusItem

### Functions

Status bar

```
StatusItemStatusBar( StatusItemRef item ) = StatusBarRef
```

Behavior

```
StatusItemButton( StatusItemRef item ) = StatusBarButtonRef// macOS 10.10+
```

```
StatusItemMenu( StatusItemRef item ) = CocoaMenuRef
```

```
StatusItemSetMenu( StatusItemRef item, CocoaMenuRef menu )
```

```
StatusItemSetMenuIndex( StatusItemRef item, NSInteger menuIndex )
```

Appearance

```
StatusItemLength( StatusItemRef item ) = CGFloat
```

```
StatusItemSetLength( StatusItemRef item, CGFloat length )
```

Instance properties

```
StatusItemAutosaveName( StatusItemRef item ) = CFStringRef// macOS 10.12+
```

```
StatusItemSetAutosaveName( StatusItemRef item, CFStringRef name )// macOS 10.12+
```

```
StatusItemBehavior( StatusItemRef item ) = NSStatusItemBehavior// macOS 10.12+
```

```
StatusItemSetBehavior( StatusItemRef item, NSStatusItemBehavior behavior )// macOS 10.12+
```

```
StatusItemIsVisible( StatusItemRef item ) = Boolean// macOS 10.12+
```

```
StatusItemSetVisible( StatusItemRef item, Boolean flag )// macOS 10.12+
```

Convenience

```
StatusItemButtonSetImage( StatusItemRef item, ImageRef image )// macOS 10.10+
```

```
StatusItemButtonAppearsDisabled( StatusItemRef item ) = Boolean// macOS 10.10+
```

```
StatusItemButtonSetAppearsDisabled( StatusItemRef item, Boolean flag )// macOS 10.10+
```

```
StatusItemButtonSetActionCallback( StatusItemRef item, ptr callback, ptr userData )// macOS 10.10+
```

### Apple documentation

[NSStatusItem](#)



## Stepper

statement

### Syntax

**stepper** *tag, enabled, value, rect, min, max, wndTag*

### Description

The **stepper** statement puts a new stepper in the current output cocoa window, or alters an existing stepper's characteristics.

### Parameters

<i>tag</i>	A number to identify the stepper. A negative tag hides the stepper.
<i>enabled</i>	Enables or disables the stepper.
<i>value</i>	The value that indicates the current extent of the stepper (default = 0).
<i>rect</i>	Origin and size of the stepper in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>min</i>	The minimum value of the stepper (default = 0).
<i>max</i>	The maximum value of the stepper (default = 100).
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

[\\_btnClick](#)

### Functions

```
StepperWithTag( NSInteger tag ) = StepperRef
```

```
StepperExists( NSInteger tag ) = Boolean
```

Init

```
StepperInit( NSInteger tag, CGRect r ) = StepperRef// autoreleased
```

Value range

```
StepperMaxValue( NSInteger tag ) = double
```

```
StepperSetMaxValue( NSInteger tag, double value )
```

```
StepperMinValue( NSInteger tag ) = double
```

```
StepperSetMinValue( NSInteger tag, double value )
```

```
StepperIncrement( NSInteger tag ) = double
```

```
StepperSetIncrement( NSInteger tag, double increment )
```

Specify how stepper responds

```
StepperAutorepeat( NSInteger tag ) = Boolean
```

```
StepperSetAutorepeat( NSInteger tag, Boolean flag )
```

```
StepperValueWraps( NSInteger tag ) = Boolean
```

```
StepperSetValueWraps( NSInteger tag, Boolean flag )
```

### See Also

[Control](#), [View](#)

### Apple documentation

[NSStepper](#)



## String

### Functions

#### Create

```
StringWithBytes( ptr bytes, NSUInteger length, NSStringEncoding encoding ) = CFStringRef
StringWithCharacters( unichar *characters, NSUInteger length ) = CFStringRef
StringWithData( CFDataRef dta, NSStringEncoding encoding ) = CFStringRef
StringWithFormat( CFStringRef format, ... ) = CFStringRef
StringWithCString( const char *cString, NSStringEncoding encoding ) = CFStringRef
StringWithUTF8String( const char *cString ) = CFStringRef
```

#### Create from URL

```
StringWithContentsOfURL( CFURLRef url, NSStringEncoding enc, ErrorRef *err ) = CFStringRef
StringWithContentsOfURLUsedEncoding( CFURLRef url, NSStringEncoding *enc, ErrorRef *err ) = CFStringRef
```

#### Length

```
StringLength( CFStringRef string ) = NSUInteger
```

#### Characters and bytes

```
StringCharacterAtIndex( CFStringRef string, NSUInteger index ) = unichar
```

#### Getting C strings

```
StringCStringUsingEncoding( CFStringRef string, NSStringEncoding enc ) = ptr// char *
StringUTF8String( CFStringRef string ) = ptr// char *
```

#### Compare

```
StringCaseInsensitiveCompare( CFStringRef string, CFStringRef otherString ) = NSComparisonResult
StringLocalizedCaseInsensitiveCompare( CFStringRef string, CFStringRef otherString ) = NSComparisonResult
StringCompare( CFStringRef string, CFStringRef otherString ) = NSComparisonResult
StringLocalizedCompare( CFStringRef string, CFStringRef otherString ) = NSComparisonResult
StringCompareWithOptions( CFStringRef string, CFStringRef otherString, NSStringCompareOptions options ) = NSComparisonResult
StringCompareWithOptionsInRange( CFStringRef string, CFStringRef otherString, NSStringCompareOptions options, CFRange range ) = NSComparisonResult
StringCompareWithOptionsInRangeWithLocale( CFStringRef string, CFStringRef otherString, NSStringCompareOptions options, CFRange range, CFLocaleRef locale ) = NSComparisonResult
StringLocalizedStandardCompare( CFStringRef string, CFStringRef otherString ) = NSComparisonResult
StringHasPrefix( CFStringRef string, CFStringRef prefix ) = Boolean
StringHasSuffix( CFStringRef string, CFStringRef suffix ) = Boolean
StringIsEqualToString( CFStringRef string, CFStringRef otherString ) = Boolean
```

#### Combining

```
StringByAppendingFormat( CFStringRef string, CFStringRef format, ... ) = CFStringRef
StringByAppendingString( CFStringRef string1, CFStringRef string2 ) = CFStringRef
StringByPaddingToLength( CFStringRef string, NSUInteger newLength, CFStringRef padString, NSUInteger padIndex ) = CFStringRef
```

#### Case

```
StringLowercaseString( CFStringRef string ) = CFStringRef
StringLocalizedLowercaseString( CFStringRef string ) = CFStringRef
StringLowercaseStringWithLocale( CFStringRef string, CFLocaleRef locale ) = CFStringRef// macOS 10.8+
StringUppercaseString( CFStringRef string ) = CFStringRef
StringLocalizedUppercaseString( CFStringRef string ) = CFStringRef
StringUppercaseStringWithLocale( CFStringRef string, CFLocaleRef locale ) = CFStringRef// macOS 10.8+
StringCapitalizedString( CFStringRef string ) = CFStringRef
StringLocalizedCapitalizedString( CFStringRef string ) = CFStringRef
StringCapitalizedStringWithLocale( CFStringRef string, CFLocaleRef locale ) = CFStringRef// macOS 10.8+
```

#### Dividing

```
StringComponentsSeparatedByString( CFStringRef string, CFStringRef separator ) = CFArrayRef
StringComponentsSeparatedByCharactersInSet( CFStringRef string, CFCharacterSetRef set ) = CFArrayRef
StringByTrimmingCharactersInSet( CFStringRef string, CFCharacterSetRef set ) = CFStringRef
StringSubstringFromIndex( CFStringRef string, CFIndex index ) = CFStringRef
StringSubstringWithRange( CFStringRef string, CFRange range ) = CFStringRef
StringSubstringToIndex( CFStringRef string, CFIndex index ) = CFStringRef
```

#### Fold

```
StringByFolding( CFStringRef string, NSStringCompareOptions options, CFLocaleRef locale ) = CFStringRef
```

#### Transform

```
StringByApplyingTransform( CFStringRef string, CFStringRef transform, Boolean reverse ) = CFStringRef// macOS 10.11+
```

## Finding

```
StringContainsString( CFStringRef string1, CFStringRef string2 ) = Boolean
StringLocalizedCaseInsensitiveContainsString( CFStringRef string1, CFStringRef string2 ) = Boolean// macOS 10.10+
StringLocalizedStandardContainsString( CFStringRef string1, CFStringRef string2 ) = Boolean// macOS 10.11+
StringRangeOfCharacterFromSet( CFStringRef string, CFCharacterSetRef set ) = CFRange
StringRangeOfCharacterFromSetWithOptions( CFStringRef string, CFCharacterSetRef set, NSStringCompareOptions
options ) = CFRange
StringRangeOfCharacterFromSetWithOptionsInRange( CFStringRef string, CFCharacterSetRef set, NSStringCompareOptions
options, CFRange range ) = CFRange
StringRangeOfString( CFStringRef string, CFStringRef searchString ) = CFRange
StringRangeOfStringWithOptions( CFStringRef string, CFStringRef searchString, NSStringCompareOptions options ) =
CFRange
StringRangeOfStringWithOptionsInRange( CFStringRef string, CFStringRef searchString, NSStringCompareOptions options,
CFRange range ) = CFRange
StringRangeOfStringWithOptionsInRangeAndLocale( CFStringRef string, CFStringRef searchString, NSStringCompareOptions
options, CFRange range, CFLocaleRef locale ) = CFRange
StringLocalizedStandardRangeOfString( CFStringRef string1, CFStringRef string2 ) = CFRange// macOS 10.11+
StringEnumerateLines( CFStringRef string, ptr callback, ptr userData )
StringEnumerateSubstringsInRange( CFStringRef string, CFRange range, NSStringEnumerationOptions options, ptr
callback, ptr userData )
```

## Replacing substrings

```
StringByReplacingOccurrencesOfString( CFStringRef string, CFStringRef target, CFStringRef replacement ) = CFStringRef
StringByReplacingOccurrencesOfStringWithOptions( CFStringRef string, CFStringRef target, CFStringRef replacement,
NSStringCompareOptions options, CFRange range ) = CFStringRef
StringByReplacingCharactersInRange( CFStringRef string, CFRange range, CFStringRef replacement ) = CFStringRef
```

## Shared prefix

```
StringCommonPrefixWithString( CFStringRef string1, CFStringRef string2, NSStringCompareOptions options ) =
CFStringRef
```

## Linguistic analysis

```
StringEnumerateLinguisticTags( CFStringRef string, CFRange range, CFStringRef scheme, NSLinguisticTaggerOptions
options, OrthographyRef orthRef, ptr callback, ptr userData )
StringLinguisticTagsInRange( CFStringRef string, CFRange range, CFStringRef scheme, NSLinguisticTaggerOptions
options, OrthographyRef orthRef, CFArrayRef *tokenRanges ) = CFArrayRef
```

## Line and paragraph ranges

```
StringGetLineStartEnd( CFStringRef string, NSUInteger *startPtr, NSUInteger *lineEndPtr, NSUInteger *contentsEndPtr,
CFRange range )
StringLineRange( CFStringRef string, CFRange range ) = CFRange
StringGetParagraphStartEnd( CFStringRef string, NSUInteger *startPtr, NSUInteger *parEndPtr, NSUInteger
*contentsEndPtr, CFRange range )
StringParagraphRange( CFStringRef string, CFRange range ) = CFRange
```

## Composed char sequences

```
StringRangeOfComposedCharacterSequenceAtIndex( CFStringRef string, NSUInteger index ) = CFRange
StringRangeOfComposedCharacterSequencesForRange( CFStringRef string, CFRange range ) = CFRange
```

## Writing to URL

```
StringWriteToURL( CFStringRef string, CFURLRef url, Boolean atomically, NSStringEncoding enc, ErrorRef *err ) =
Boolean
```

## Property list

```
StringPropertyList( CFStringRef string ) = CFTypeRef
StringPropertyListFromStringsFileFormat( CFStringRef string ) = CFDictionaryRef
```

## Sizing and drawing

```
StringDrawAtPoint( CFStringRef string, CGPoint pt, CFDictionaryRef attributes )
StringDrawInRect( CFStringRef string, CGRect rect, CFDictionaryRef attributes )
StringDrawWithRect( CFStringRef string, CGRect rect, NSStringDrawingOptions options, CFDictionaryRef attributes,
StringDrawingContextRef context )// macOS 10.11+
StringBoundingRectWithSize( CFStringRef string, CGSize size, NSStringDrawingOptions options, CFDictionaryRef
attributes, StringDrawingContextRef context ) = CGRect// macOS 10.11+
StringSizeWithAttributes( CFStringRef string, CFDictionaryRef attributes ) = CGSize
```

## Numeric values

```
StringDoubleValue( CFStringRef string ) = double
StringFloatValue( CFStringRef string ) = float
StringIntValue( CFStringRef string ) = SInt32
StringIntegerValue( CFStringRef string ) = NSInteger
StringLongLongValue( CFStringRef string ) = SInt64
StringBoolValue( CFStringRef string ) = Boolean
```

## Working with encodings

```
StringEncodingForData( CFDataRef dta, CFDictionaryRef encodingOptions, CFStringRef *convertedString, Boolean
*usedLossyConversion ) = NSStringEncoding// macOS 10.10+
StringLocalizedNameOfStringEncoding( NSStringEncoding encoding ) = CFStringRef
StringCanBeConvertedToEncoding( CFStringRef string, NSStringEncoding encoding ) = Boolean
StringData( CFStringRef string, NSStringEncoding encoding ) = CFDataRef
StringDataAllowLossyConversion( CFStringRef string, NSStringEncoding encoding, Boolean lossy ) = CFDataRef
```

```
StringFastestEncoding( CFStringRef string ) = NSStringEncoding
StringSmallestEncoding( CFStringRef string ) = NSStringEncoding
```

#### Paths

```
StringPathWithComponents( CFArrayRef array ) = CFStringRef
StringPathComponents( CFStringRef string ) = CFArrayRef
StringIsAbsolutePath( CFStringRef string ) = Boolean
StringLastPathComponent( CFStringRef string ) = CFStringRef
StringPathExtension( CFStringRef string ) = CFStringRef
StringByAbbreviatingWithTildeInPath( CFStringRef string ) = CFStringRef
StringByAppendingPathComponent( CFStringRef string, CFStringRef component ) = CFStringRef
StringByAppendingPathExtension( CFStringRef string, CFStringRef extension ) = CFStringRef
StringByDeletingLastPathComponent( CFStringRef string ) = CFStringRef
StringByDeletingPathExtension( CFStringRef string ) = CFStringRef
StringByExpandingTildeInPath( CFStringRef string ) = CFStringRef
StringByResolvingSymlinksInPath( CFStringRef string ) = CFStringRef
StringByStandardizingPath( CFStringRef string ) = CFStringRef
StringStringsByAppendingPaths( CFStringRef string, CFArrayRef paths ) = CFArrayRef
```

#### URL strings

```
StringByAddingPercentEncodingWithAllowedCharacters( CFStringRef string, CFCharacterSetRef allowedCharacters ) =
CFStringRef// macOS 10.9+
StringByRemovingPercentEncoding( CFStringRef string ) = CFStringRefmacOS 10.9+
```

#### • Mutable string •

```
StringWithCapacity( NSUInteger capacity ) = CFMutableStringRef
StringAppendFormat( CFMutableStringRef string, CFStringRef format, ... )
StringAppendString( CFMutableStringRef string, CFStringRef otherString )
StringApplyTransform( CFMutableStringRef string, NSStringTransform transform, Boolean reverse, CFRange range,
CFRange *resultingRange )
StringDeleteCharacters( CFMutableStringRef string, CFRange range )
StringInsertString( CFMutableStringRef string, CFStringRef otherString, NSUInteger index )
StringReplaceCharacters( CFMutableStringRef string, CFStringRef otherString, CFRange range )
StringReplaceOccurrencesOfString( CFMutableStringRef string, CFStringRef targetString, CFStringRef
replacementString, NSStringCompareOptions options, CFRange range )
StringSetString( CFMutableStringRef string, CFStringRef otherString )
```

#### Apple documentation

[NSString](#)

[NSMutableString](#)





---

### StringDrawingContext

#### Functions

Scale factors

```
StringDrawingContextMinimumScaleFactor( StringDrawingContextRef ctx ) = CGFloat  
StringDrawingContextSetMinimumScaleFactor( StringDrawingContextRef ctx, CGFloat factor )  
StringDrawingContextActualScaleFactor( StringDrawingContextRef ctx ) = CGFloat
```

Bounds

```
StringDrawingContextTotalBounds( StringDrawingContextRef ctx ) = CGRect
```

#### Apple documentation

[NSStringDrawingContext](#)



---

### Subclass

statement

#### Description

Placing the **subclass** keyword next to a widget statement subclasses the widget. Subclassed widgets trigger additional user dialog events. For example, a subclassed view triggers `_viewMouseDown`, `_viewKeyDown`, `_viewDraggingEntered`, etc. events.

N.B. Only use the **subclass** keyword when initially building a window or widget. Do not use it when modifying a window or widget.



---

## TableRowView

### Functions

Init

```
TableRowViewInit = TableRowViewRef// autoreleased
```

Display style

```
TableRowViewIsEmphasized( TableRowViewRef ref ) = Boolean  
TableRowViewSetEmphasized( TableRowViewRef ref, Boolean flag )  
TableRowViewInteriorBackgroundStyle( TableRowViewRef ref ) = NSBackgroundStyle  
TableRowViewIsFloating( TableRowViewRef ref ) = Boolean  
TableRowViewSetFloating( TableRowViewRef ref, Boolean flag )
```

Row selection

```
TableRowViewIsSelected( TableRowViewRef ref ) = Boolean  
TableRowViewSetSelected( TableRowViewRef ref, Boolean flag )  
TableRowViewSelectionHighlightStyle( TableRowViewRef ref ) = NSTableViewSelectionHighlightStyle  
TableRowViewSetSelectionHighlightStyle( TableRowViewRef ref, NSTableViewSelectionHighlightStyle style )
```

Row grouping

```
TableRowViewIsGroupRowStyle( TableRowViewRef ref ) = Boolean  
TableRowViewSetGroupRowStyle( TableRowViewRef ref, Boolean flag )  
TableRowViewNumberOfColumns( TableRowViewRef ref ) = NSInteger
```

Overriding display characteristics

```
TableRowViewBackgroundColor( TableRowViewRef ref ) = ColorRef  
TableRowViewSetBackgroundColor( TableRowViewRef ref, ColorRef col )
```

Row column view

```
TableRowViewViewAtColumn( TableRowViewRef ref, NSInteger column ) = CFTypeRef
```

Instance properties

```
TableRowViewIsNextRowSelected( TableRowViewRef ref ) = Boolean// macOS 10.10+  
TableRowViewSetNextRowSelected( TableRowViewRef ref, Boolean flag )// macOS 10.10+  
TableRowViewIsPreviousRowSelected( TableRowViewRef ref ) = Boolean// macOS 10.10+  
TableRowViewSetPreviousRowSelected( TableRowViewRef ref, Boolean flag )// macOS 10.10+
```

### Apple documentation

[NSTableView](#)



## TableView

### Description

The **tableView** must be view-based and created in a nib window. Requires the Cocoa runtime ([CocoaInit](#)).

### Dialog Events

`_tableViewDoubleAction`  
`_tableViewSelectionDidChange`

### Functions

`TableViewWithTag( NSInteger tag ) = TableViewRef`  
`TableViewExists( NSInteger tag ) = Boolean`

#### Init

`TableViewInit( NSInteger tag, CGRect r ) = TableViewRef// autoreleased`

#### Data source

`TableViewUsesStaticContents( NSInteger tag ) = Boolean`  
`TableViewSetUsesStaticContents( NSInteger tag, Boolean flag )`  
`TableViewSetData( NSInteger tag, CFMutableArrayRef array )`  
`TableViewData( NSInteger tag ) = CFMutableArrayRef`  
`TableViewReloadData( NSInteger tag )`  
`TableViewReloadDataForRowsColumns( NSInteger tag, IndexSetRef rows, IndexSetRef columns )`

#### Update

`TableViewBeginUpdates( NSInteger tag )`  
`TableViewEndUpdates( NSInteger tag )`  
`TableViewMoveRow( NSInteger tag, NSInteger atIndex, NSInteger toIndex )`

#### Target-action

`TableViewClickedRow( NSInteger tag ) = NSInteger`  
`TableViewClickedColumn( NSInteger tag ) = NSInteger`

#### Behavior

`TableViewAllowsColumnReordering( NSInteger tag ) = Boolean`  
`TableViewSetAllowsColumnReordering( NSInteger tag, Boolean flag )`  
`TableViewAllowsColumnResizing( NSInteger tag ) = Boolean`  
`TableViewSetAllowsColumnResizing( NSInteger tag, Boolean flag )`  
`TableViewAllowsMultipleSelection( NSInteger tag ) = Boolean`  
`TableViewSetAllowsMultipleSelection( NSInteger tag, Boolean flag )`  
`TableViewAllowsEmptySelection( NSInteger tag ) = Boolean`  
`TableViewSetAllowsEmptySelection( NSInteger tag, Boolean flag )`  
`TableViewAllowsColumnSelection( NSInteger tag ) = Boolean`  
`TableViewSetAllowsColumnSelection( NSInteger tag, Boolean flag )`  
`TableViewUsesAutomaticRowHeights( NSInteger tag ) = Boolean// macOS 10.13+`  
`TableViewSetUsesAutomaticRowHeights( NSInteger tag, Boolean flag )// macOS 10.13+`

#### Display attributes

`TableViewIntercellSpacing( NSInteger tag ) = CGSize`  
`TableViewSetIntercellSpacing( NSInteger tag, CGSize spacing )`  
`TableViewRowHeight( NSInteger tag ) = CGFloat`  
`TableViewSetRowHeight( NSInteger tag, CGFloat height )`  
`TableViewBackgroundColor( NSInteger tag ) = ColorRef`  
`TableViewSetBackgroundColor( NSInteger tag, ColorRef col )`  
`TableViewUsesAlternatingRowBackgroundColors( NSInteger tag ) = Boolean`  
`TableViewSetUsesAlternatingRowBackgroundColors( NSInteger tag, Boolean flag )`  
`TableViewSelectionHighlightStyle( NSInteger tag ) = NSTableViewSelectionHighlightStyle`  
`TableViewSetSelectionHighlightStyle( NSInteger tag, NSTableViewSelectionHighlightStyle style )`  
`TableViewGridColor( NSInteger tag ) = ColorRef`  
`TableViewSetGridColor( NSInteger tag, ColorRef col )`  
`TableViewGridStyleMask( NSInteger tag ) = NSTableViewGridLineStyle`  
`TableViewSetGridStyleMask( NSInteger tag, NSTableViewGridLineStyle style )`  
`TableViewIndicatorImageInTableColumn( NSInteger tag, TableColumnRef columnRef ) = ImageRef`  
`TableViewSetIndicatorImageInTableColumn( NSInteger tag, ImageRef image, TableColumnRef columnRef )`

#### Row size styles

`TableViewEffectiveRowSizeStyle( NSInteger tag ) = NSTableViewRowSizeStyle`  
`TableViewRowSizeStyle( NSInteger tag ) = NSTableViewRowSizeStyle`  
`TableViewSetRowSizeStyle( NSInteger tag, NSTableViewRowSizeStyle style )`

#### Column management

```
TableViewMoveColumn( NSInteger tag, NSInteger atIndex, NSInteger toIndex )
TableViewTableColumns( NSInteger tag ) = CFArrayRef
TableViewColumnWithIdentifier( NSInteger tag, CFStringRef identifier ) = NSInteger
TableViewTableColumnWithIdentifier( NSInteger tag, CFStringRef identifier ) = TableColumnRef
```

#### Selecting columns and rows

```
TableViewSelectColumns( NSInteger tag, IndexSetRef columnIndexes, Boolean extendSelection )
TableViewSelectedColumn( NSInteger tag ) = NSInteger
TableViewSelectedColumns( NSInteger tag ) = IndexSetRef
TableViewDeselectColumn( NSInteger tag, NSInteger columnIndex )
TableViewNumberOfSelectedColumns( NSInteger tag ) = NSInteger
TableViewIsColumnSelected( NSInteger tag, NSInteger columnIndex ) = Boolean
TableViewSelectRows( NSInteger tag, IndexSetRef rowIndexes, Boolean extendSelection )
TableViewSelectRow( NSInteger tag, NSInteger rowIndex )
TableViewSelectedRow( NSInteger tag ) = NSInteger
TableViewSelectedRowIndexes( NSInteger tag ) = CFArrayRef
TableViewDeselectRow( NSInteger tag, NSInteger row )
TableViewNumberOfSelectedRows( NSInteger tag ) = NSInteger
TableViewIsRowSelected( NSInteger tag, NSInteger row ) = Boolean
TableViewSelectAll( NSInteger tag )
TableViewDeselectAll( NSInteger tag )
```

#### Type select

```
TableViewAllowsTypeSelect( NSInteger tag ) = Boolean
TableViewSetAllowsTypeSelect( NSInteger tag, Boolean flag )
```

#### Dimensions

```
TableViewNumberOfColumns( NSInteger tag ) = NSInteger
```

#### Floating rows

```
TableViewFloatsGroupRows( NSInteger tag ) = Boolean
TableViewSetFloatsGroupRows( NSInteger tag, Boolean flag )
```

#### Editing cells

```
TableViewEditColumnRow( NSInteger tag, NSInteger columnIndex, NSInteger row, Boolean selectFlag )
```

#### Scrolling

```
TableViewScrollRowToVisible( NSInteger tag, NSInteger row )
TableViewScrollColumnToVisible( NSInteger tag, NSInteger col )
```

#### Column state persistence

```
TableViewAutosaveColumns( NSInteger tag ) = Boolean
TableViewSetAutosaveColumns( NSInteger tag, Boolean flag )
TableViewAutosaveName( NSInteger tag ) = CFStringRef
TableViewSetAutosaveName( NSInteger tag, CFStringRef name )
```

#### Highlightable column headers

```
TableViewHighlightedTableColumn( NSInteger tag ) = TableColumnRef
TableViewSetHighlightedTableColumn( NSInteger tag, TableColumnRef columnRef )
```

#### Dragging

```
TableViewCanDragRows( NSInteger tag, IndexSetRef rowIndexes, CGPoint atPoint ) = Boolean
TableViewSetDraggingSourceOperationMask( NSInteger tag, NSDragOperation dragOperation, Boolean forLocal )
TableViewVerticalMotionCanBeginDrag( NSInteger tag ) = Boolean
TableViewSetVerticalMotionCanBeginDrag( NSInteger tag, Boolean flag )
TableViewDraggingDestinationFeedbackStyle( NSInteger tag ) = NSTableViewDraggingDestinationFeedbackStyle
TableViewSetDraggingDestinationFeedbackStyle( NSInteger tag, NSTableViewDraggingDestinationFeedbackStyle style )
TableViewSetDropRow( NSInteger tag, NSInteger row, NSTableViewDropOperation operation )
```

#### Sorting

```
TableViewSortDescriptors( NSInteger tag ) = CFArrayRef
TableViewSetSortDescriptors( NSInteger tag, CFArrayRef descriptors )
```

#### Row actions

```
TableViewRowActionsVisible( NSInteger tag ) = Boolean// macOS 10.11+
TableViewSetRowActionsVisible( NSInteger tag, Boolean flag )// macOS 10.11+
```

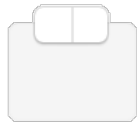
#### Hiding and showing rows

```
TableViewHideRow( NSInteger tag, NSInteger rowIndex, NSTableViewAnimationOptions animation )macOS 10.11+
TableViewHideRows( NSInteger tag, IndexSetRef indexes, NSTableViewAnimationOptions animation )macOS 10.11+
TableViewUnhideRow( NSInteger tag, NSInteger rowIndex, NSTableViewAnimationOptions animation )macOS 10.11+
TableViewUnhideRows( NSInteger tag, IndexSetRef indexes, NSTableViewAnimationOptions animation )macOS 10.11+
TableViewHiddenRowIndexes( NSInteger tag ) = IndexSetRefmacOS 10.11+
```

#### Convenience

```
TableViewView( NSInteger tag, NSInteger row, NSInteger col ) = ptr
TableViewSetDelegateCallback( NSInteger tag, ptr callback )
TableViewCellMakeFirstResponder( NSInteger tag, NSInteger row, NSInteger col )
toolbox TableViewSetHeaderFont( NSInteger tag, FontRef font )// macOS 10.10+
toolbox TableViewSetHeaderTextColor( NSInteger tag, ColorRef txtColor )// macOS 10.10+
```





TabView

statement

Syntax

**tabview** *tag, index, items, rect, wndTag*

Description

The **tabview** statement puts a new tabview in the current output cocoa window, or alters an existing tabview’s characteristics.

Parameters

<i>tag</i>	A number to identify the tabview. A negative tag hides the tabview.
<i>index</i>	The index of the currently selected item (default = 0).
<i>items</i>	A CFArray or semicolon-delimited string of item titles.
<i>rect</i>	Origin and size of the tabview in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

Dialog Events

Event	Description
<i>_tabViewDidChangeNumberOfTabViewItems</i>	The number of tabview items have changed.
<i>_tabViewShouldSelectTabViewItem</i>	Invoked just before the tabview item is selected.
<i>_tabViewWillSelectTabViewItem</i>	The tabview item is about to be selected.
<i>_tabViewDidSelectTabViewItem</i>	The tabview item has been selected.

Functions

`TabViewWithTag( NSInteger tag ) = TabViewRef`  
`TabViewExists( NSInteger tag ) = Boolean`

Init  
`TabViewInit( NSInteger tag, CGRect r ) = TabViewRef// autoreleased`

Accessing tabs  
`TabViewIndex( NSInteger tag ) = NSInteger`

Selecting tabs  
`TabViewSelectFirstItem( NSInteger tag )`  
`TabViewSelectLastItem( NSInteger tag )`  
`TabViewSelectNextItem( NSInteger tag )`  
`TabViewSelectPreviousItem( NSInteger tag )`

Font  
`TabViewSetFont( NSInteger tag, CTNSFontRef font )`

Type  
`TabViewType( NSInteger tag ) = NSTabViewType`  
`TabViewSetType( NSInteger tag, NSInteger type ) = NSInteger`

Control tint  
`TabViewControlTint( NSInteger tag ) = NSControlTint`  
`TabViewSetControlTint( NSInteger tag, NSControlTint tint )`

Background  
`TabViewDrawsBackground( NSInteger tag ) = Boolean`  
`TabViewSetDrawsBackground( NSInteger tag, Boolean flag )`

Determining size  
`TabViewMinimumSize( NSInteger tag ) = CGSize`

```
TabViewContentRect( NSInteger tag ) = CGRect
TabViewControlSize( NSInteger tag ) = CGSize
```

Truncating labels

```
TabViewAllowsTruncatedLabels( NSInteger tag ) = Boolean
TabViewSetAllowsTruncatedLabels( NSInteger tag, Boolean flag )
```

Instance properties

```
TabViewPosition( NSInteger tag ) = NSTabPosition// macOS 10.12+
TabViewSetPosition( NSInteger tag, NSTabPosition position )// macOS 10.12+
TabViewBorderType( NSInteger tag ) = NSTabViewBorderType// macOS 10.12+
TabViewSetBorderType( NSInteger tag, NSTabViewBorderType position )// macOS 10.12+
```

Convenience

```
TabViewAddSubview( NSInteger tabTag, NSInteger index, NSInteger subviewTag )
```

## See Also

[Control](#), [View](#)

## Apple documentation

[NSTabView](#)





## Task

---

### Functions

Init

```
TaskInit = TaskRef// autoreleased  
TaskLaunchedTaskWithExecutableURL( CFURLRef url, CFArrayRef arguments, ErrorRef *err, ptr terminationHandler ) =  
TaskRef// macOS 10.13+
```

Info and configure

```
TaskArguments( TaskRef tsk ) = CFArrayRef  
TaskSetArguments( TaskRef tsk, CFArrayRef arguments )  
TaskCurrentDirectoryURL( TaskRef tsk ) = CFURLRef  
TaskSetCurrentDirectoryURL( TaskRef ts, CFURLRef url )  
TaskExecutableURL( TaskRef tsk ) = CFURLRef  
TaskSetExecutableURL( TaskRef ts, CFURLRef url )  
TaskEnvironment( TaskRef tsk ) = CFDictionaryRef  
TaskSetEnvironment( TaskRef tsk, CFDictionaryRef environment )  
TaskProcessIdentifier( TaskRef tsk ) = long  
TaskStandardError( TaskRef tsk ) = ptr// ptr = FileHandleRef or PipeRef  
TaskSetStandardError( TaskRef tsk, ptr obj )// ptr = FileHandleRef or PipeRef  
TaskStandardInput( TaskRef tsk ) = ptr// ptr = FileHandleRef or PipeRef  
TaskSetStandardInput( TaskRef tsk, ptr obj )// ptr = FileHandleRef or PipeRef  
TaskStandardOutput( TaskRef tsk ) = ptr// ptr = FileHandleRef or PipeRef  
TaskSetStandardOutput( TaskRef tsk, ptr obj )// ptr = FileHandleRef or PipeRef
```

Running and stopping

```
TaskInterrupt( TaskRef tsk )  
TaskLaunch( TaskRef tsk, ErrorRef *err ) = Boolean// Note: the return value and err param are only used on macOS  
10.13+  
TaskResume( TaskRef tsk ) = Boolean  
TaskSuspend( TaskRef tsk ) = Boolean  
TaskTerminate( TaskRef tsk )  
TaskWaitUntilExit( TaskRef tsk )
```

Query state

```
TaskIsRunning( TaskRef tsk ) = Boolean  
TaskTerminationStatus( TaskRef tsk ) = long  
TaskTerminationReason( TaskRef tsk ) = NSTaskTerminationReason
```

Termination handler

```
TaskSetTerminationHandler( TaskRef tsk, ptr handlerFn )
```

### Apple documentation

[NSTask](#)



## Text

### Description

Generic functions and dialog events for descendants of the text class (e.g. textView).

### Dialog Events

```
_textDidBeginEditing  
_textDidChange  
_textDidEndEditing
```

### Functions

Getting characters

```
TextString( NSInteger tag ) = CFStringRef  
TextSetString( NSInteger tag, CFStringRef string )
```

Changing the selection

```
TextSelectedRange( NSInteger tag ) = CFRange  
TextSetSelectedRange( NSInteger tag, CFRange range )
```

Replacing text

```
TextReplaceCharactersInRangeWithString( NSInteger tag, CFRange range, CFStringRef string )
```

Action methods for editing

```
TextSelectAll( NSInteger tag )  
TextCopy( NSInteger tag )  
TextCut( NSInteger tag )  
TextPaste( NSInteger tag )  
TextCopyFont( NSInteger tag )  
TextPasteFont( NSInteger tag )  
TextDelete( NSInteger tag )
```

Font

```
TextSetFont( NSInteger tag, CTNSFontRef font )  
TextSetFontInRange( NSInteger tag, CTNSFontRef font, CFRange range )  
TextSetFontWithName( NSInteger tag, CFStringRef name, CGFloat size )  
TextSetFontWithNameInRange( NSInteger tag, CFStringRef name, CGFloat size, CFRange range )
```

Text alignment

```
TextAlignment( NSInteger tag ) = NSTextAlignment  
TextSetAlignment( NSInteger tag, NSTextAlignment alignment )  
TextAlignCenter( NSInteger tag )  
TextAlignLeft( NSInteger tag )  
TextAlignRight( NSInteger tag )
```

Text color

```
TextColor( NSInteger tag ) = ColorRef  
TextSetColor( NSInteger tag, ColorRef col )  
TextSetColorInRange( NSInteger tag, ColorRef col, CFRange range )
```

Superscript and subscript

```
TextSuperscript( NSInteger tag )  
TextSubscript( NSInteger tag )  
TextUnscript( NSInteger tag )
```

Underline

```
TextUnderline( NSInteger tag )
```

Reading and writing RTF files

```
TextReadRTFDFromURL( NSInteger tag, CFURLRef url ) = Boolean  
TextWriteRTFDToURL( NSInteger tag, CFURLRef url, Boolean atomically ) = Boolean
```

Spelling

```
TextShowGuessPanel( NSInteger tag )
```

Constraining size

```
TextMaxSize( NSInteger tag ) = CGSize  
TextSetMaxSize( NSInteger tag, CGSize size )  
TextMinSize( NSInteger tag ) = CGSize  
TextSetMinSize( NSInteger tag, CGSize size )  
TextIsVerticallyResizable( NSInteger tag ) = Boolean  
TextSetVerticallyResizable( NSInteger tag, Boolean flag )
```

```
TextIsHorizontallyResizable( NSInteger tag ) = Boolean  
TextSetHorizontallyResizable( NSInteger tag, Boolean flag )  
TextSizeToFit( NSInteger tag )
```

Scrolling

```
TextScrollRangeToVisible( NSInteger tag, CFRange range )
```

## Apple documentation

[NSText](#)



---

## TextAttachment

### Functions

#### Init

```
TextAttachmentInit = TextAttachmentRef// autoreleased
TextAttachmentWithFileWrapper( FileWrapperRef fw ) = TextAttachmentRef
TextAttachmentWithData( CFDataRef dta, CFStringRef type ) = TextAttachmentRef// macOS 10.11+
```

#### Contents

```
TextAttachmentBounds( TextAttachmentRef ref ) = CGRect// -m64, macOS 10.11+
TextAttachmentSetBounds( TextAttachmentRef ref, CGRect r )// -m64, macOS 10.11+
TextAttachmentContents( TextAttachmentRef ref ) = CFDataRef// -m64, macOS 10.11+
TextAttachmentSetContents( TextAttachmentRef ref, CFDataRef dta )// -m64, macOS 10.11+
TextAttachmentFileType( TextAttachmentRef ref ) = CFStringRef// -m64, macOS 10.11+
TextAttachmentSetFileType( TextAttachmentRef ref, CFStringRef type )// -m64, macOS 10.11+
TextAttachmentImage( TextAttachmentRef ref ) = ImageRef// -m64, macOS 10.11+
TextAttachmentSetImage( TextAttachmentRef ref, ImageRef image )// -m64, macOS 10.11+
TextAttachmentFileWrapper( TextAttachmentRef ref ) = FileWrapperRef
TextAttachmentSetFileWrapper( TextAttachmentRef ref, FileWrapperRef wrapper )
```

#### Cell

```
TextAttachmentCell( TextAttachmentRef ref ) = TextAttachmentCellRef
TextAttachmentSetCell( TextAttachmentRef ref, TextAttachmentCellRef cell )
```

#### - TextAttachmentCell -

##### Init

```
TextAttachmentCellWithImage( ImageRef image ) = TextAttachmentCellRef
```

##### Draw

```
TextAttachmentCellDraw( TextAttachmentCellRef ref, CGRect frame, NSInteger controlViewTag )
TextAttachmentCellDrawCharacterIndex( TextAttachmentCellRef ref, CGRect frame, NSInteger controlViewTag, NSUInteger charIndex )
TextAttachmentCellDrawCharacterIndexLayoutManager( TextAttachmentCellRef ref, CGRect frame, NSInteger controlViewTag, NSUInteger charIndex, LayoutManagerRef layoutManager )
TextAttachmentCellHighlight( TextAttachmentCellRef ref, Boolean highlight, CGRect frame, NSInteger controlViewTag )
```

##### Cell size and position

```
TextAttachmentCellSize( TextAttachmentCellRef ref ) = CGSize
TextAttachmentCellBaselineOffset( TextAttachmentCellRef ref ) = CGPoint
TextAttachmentCellFrameForTextContainer( TextAttachmentCellRef ref, TextContainerRef tc, CGRect proposedLineFragment, CGPoint glyphPosition, NSUInteger charIndex ) = CGRect
```

##### Set attachment

```
TextAttachmentCellSetAttachment( TextAttachmentCellRef ref, TextAttachmentRef attachment )
```

### Apple documentation

[NSTextAttachment](#)

[NSTextAttachmentCell](#)



---

## TextBlock

### Functions

Create

```
TextBlockInit = TextBlockRef// autoreleased
```

Dimensions

```
TextBlockSetValue( TextBlockRef ref, CGFloat value, NSTextBlockValueType type, NSTextBlockDimension dimension )
TextBlockValueForDimension( TextBlockRef ref, NSTextBlockDimension dimension ) = CGFloat
TextBlockValueTypeForDimension( TextBlockRef ref, NSTextBlockDimension dimension ) = NSTextBlockValueType
TextBlockContentWidth( TextBlockRef ref ) = CGFloat
TextBlockContentWidthValueType( TextBlockRef ref ) = NSTextBlockValueType
```

Margins, borders and padding

```
TextBlockSetWidthEdge( TextBlockRef ref, CGFloat value, NSTextBlockValueType type, NSTextBlockLayer layer,
CGRectEdge edge )
TextBlockSetWidth( TextBlockRef ref, CGFloat value, NSTextBlockValueType type, NSTextBlockLayer layer )
TextBlockWidthForLayer( TextBlockRef ref, NSTextBlockLayer layer, CGRectEdge edge ) = CGFloat
TextBlockWidthValueTypeForLayer( TextBlockRef ref, NSTextBlockLayer layer, CGRectEdge edge ) = NSTextBlockValueType
```

Alignment

```
TextBlockVerticalAlignment( TextBlockRef ref ) = NSTextBlockVerticalAlignment
TextBlockSetVerticalAlignment( TextBlockRef ref, NSTextBlockVerticalAlignment alignment )
```

Color

```
TextBlockBackgroundColor( TextBlockRef ref ) = ColorRef
TextBlockSetBackgroundColor( TextBlockRef ref, ColorRef col )
TextBlockSetBorderColorForEdge( TextBlockRef ref, ColorRef col, CGRectEdge edge )
TextBlockSetBorderColor( TextBlockRef ref, ColorRef col )
TextBlockBorderColorForEdge( TextBlockRef ref, CGRectEdge edge ) = ColorRef
```

Size and position

```
TextBlockRectForLayoutAtPoint( TextBlockRef ref, CGPoint pt, CGRect inRect, TextContainerRef textContainer, CFRange
charRange ) = CGRect
TextBlockBoundsRectForContentRect( TextBlockRef ref, CGRect contentRect, CGRect inRect, TextContainerRef
textContainer, CFRange charRange ) = CGRect
```

Drawing

```
TextBlockDrawBackgroundWithFrame( TextBlockRef ref, CGRect frame, ViewRef controlView, CFRange charRange,
LayoutManagerRef layoutManager )
```

Instance methods

```
TextBlockSetContentWidth( TextBlockRef ref, CGFloat width, NSTextBlockValueType type )
```

### Apple documentation

[NSTextBlock](#)



## TextCheckingResult

---

### Functions

Range and type

```
TextCheckingResultRange( TextCheckingResultRef ref ) = CFRange
TextCheckingResultType( TextCheckingResultRef ref ) = NSTextCheckingType
TextCheckingResultNumberOfRanges( TextCheckingResultRef ref ) = NSUInteger
TextCheckingResultRangeAtIndex( TextCheckingResultRef ref, NSUInteger index ) = CFRange
```

Text replacement

```
TextCheckingResultReplacementCheckingResult( CFRange range, CFStringRef replacementString ) = TextCheckingResultRef
TextCheckingResultReplacementString( TextCheckingResultRef ref ) = CFStringRef
```

Regular expressions

```
TextCheckingResultRegularExpressionCheckingResult( CFRange *ranges, NSUInteger count, RegularExpressionRef regex ) =
TextCheckingResultRef
TextCheckingResultRegularExpression( TextCheckingResultRef ref ) = RegularExpressionRef
```

Components

```
TextCheckingResultComponents( TextCheckingResultRef ref ) = CFDictionaryRef
```

URLs

```
TextCheckingResultLinkCheckingResult( CFRange range, CFURLRef url ) = TextCheckingResultRef
TextCheckingResultURL( TextCheckingResultRef ref ) = CFURLRef
```

Addresses

```
TextCheckingResultAddressCheckingResult( CFRange range, CFDictionaryRef components ) = TextCheckingResultRef
TextCheckingResultAddressComponents( TextCheckingResultRef ref ) = CFDictionaryRef
```

Transit info

```
TextCheckingResultTransitInformationCheckingResult( CFRange range, CFDictionaryRef components ) =
TextCheckingResultRef
```

Phone numbers

```
TextCheckingResultPhoneNumberCheckingResult( CFRange range, CFStringRef phoneNumber ) = TextCheckingResultRef
TextCheckingResultPhoneNumber( TextCheckingResultRef ref ) = CFStringRef
```

Dates and times

```
TextCheckingResultDateCheckingResult( CFRange range, CFDateRef dt ) = TextCheckingResultRef
TextCheckingResultDateCheckingResultWithTimeZone( CFRange range, CFDateRef dt, CFTimeZoneRef zone, CFTimeInterval
duration ) = TextCheckingResultRef
TextCheckingResultDate( TextCheckingResultRef ref ) = CFDateRef
TextCheckingResultDuration( TextCheckingResultRef ref ) = CFTimeInterval
TextCheckingResultTimeZone( TextCheckingResultRef ref ) = CFTimeZoneRef
```

Typography

```
TextCheckingResultDashCheckingResult( CFRange range, CFStringRef replacementString ) = TextCheckingResultRef
TextCheckingResultQuoteCheckingResult( CFRange range, CFStringRef replacementString ) = TextCheckingResultRef
```

Spelling

```
TextCheckingResultSpellCheckingResult( CFRange range ) = TextCheckingResultRef
TextCheckingResultCorrectionCheckingResult( CFRange range, CFStringRef replacementString ) = TextCheckingResultRef
```

Orthography

```
TextCheckingResultOrthographyCheckingResult( CFRange range, OrthographyRef orthography ) = TextCheckingResultRef
TextCheckingResultOrthography( TextCheckingResultRef ref ) = OrthographyRef
```

Grammar

```
TextCheckingResultGrammarCheckingResult( CFRange range, CFArrayRef details ) = TextCheckingResultRef
TextCheckingResultGrammarDetails( TextCheckingResultRef ref ) = CFArrayRef
```

Adjust ranges

```
TextCheckingResultByAdjustingRangesWithOffset( TextCheckingResultRef ref, NSInteger offset ) = TextCheckingResultRef
```

Instance properties

```
TextCheckingResultAlternativeStrings( TextCheckingResultRef ref ) = CFArrayRef// macOS 10.9+
```

Instance methods

```
TextCheckingResultRangeWithName( TextCheckingResultRef ref, CFStringRef name ) = CFRange// macOS 10.13+
```

Type methods

```
TextCheckingResultCorrectionCheckingResultWithAlternativeStrings( CFRange range, CFStringRef replacementString,  
CFArrayRef alternativeStrings ) = TextCheckingResultRef// macOS 10.9+
```

## Apple documentation

[TextCheckingResult](#)



---

## TextContainer

### Functions

Create  
`TextContainerWithSize( CGSize size ) = TextContainerRef` // macOS 10.11+ // autoreleased

#### Components

```
TextContainerLayoutManager( TextContainerRef ref ) = LayoutManagerRef
TextContainerSetLayoutManager( TextContainerRef ref, LayoutManagerRef lm )
TextContainerReplaceLayoutManager( TextContainerRef ref, LayoutManagerRef lm )
TextContainerTextView( TextContainerRef tc ) = NSInteger
```

#### Shape

```
TextContainerSize( TextContainerRef tc ) = CGSize // macOS 10.11+
TextContainerSetSize( TextContainerRef tc, CGSize size ) // macOS 10.11+
TextContainerExclusionPaths( TextContainerRef tc ) = CFArrayRef // macOS 10.11+ // array of BezierPaths
TextContainerSetExclusionPaths( TextContainerRef tc, CFArrayRef paths ) // macOS 10.11+ // array of BezierPaths
TextContainerLineBreakMode( TextContainerRef tc ) = NSInteger // macOS 10.11+
TextContainerSetLineBreakMode( TextContainerRef tc, NSInteger lineBreakMode ) // macOS 10.11+
TextContainerWidthTracksTextView( TextContainerRef tc ) = Boolean
TextContainerSetWidthTracksTextView( TextContainerRef tc, Boolean flag )
TextContainerHeightTracksTextView( TextContainerRef tc ) = Boolean
TextContainerSetHeightTracksTextView( TextContainerRef tc, Boolean flag )
```

#### Constrain

```
TextContainerMaximumNumberOfLines( TextContainerRef tc ) = NSInteger // macOS 10.11+
TextContainerSetMaximumNumberOfLines( TextContainerRef tc, NSInteger lines ) // macOS 10.11+
TextContainerLineFragmentPadding( TextContainerRef tc ) = CGFloat
TextContainerSetLineFragmentPadding( TextContainerRef tc, CGFloat padding )
TextContainerLineFragmentRectForProposedRect( TextContainerRef tc, CGRect proposedRect, NSInteger index,
NSWritingDirection direction, CGRect *remainingRect ) = CGRect // macOS 10.11+
TextContainerIsSimpleRectangularTextContainer( TextContainerRef tc ) = Boolean
```

### Apple documentation

[NSTextContainer](#)





## TextField

statement

### Syntax

**textfield** *tag, enabled, text, rect, wndTag*

### Description

The **textfield** statement puts a new textfield in the current output cocoa window, or alters an existing textfield's characteristics.

### Parameters

<i>tag</i>	A number to identify the textfield. A negative tag hides the textfield.
<i>enabled</i>	Enables or disables the textfield.
<i>text</i>	The textfield's text.
<i>rect</i>	Origin and size of the textfield in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

Event	Description
<i>_textFieldDidBeginEditing</i>	Sent at the beginning of an editing session.
<i>_textFieldDidChange</i>	Sent when the text changes.
<i>_textFieldDidEndEditing</i>	Sent at the end of an editing session.

### Functions

See [control](#)

```
TextFieldWithTag( NSInteger tag ) = TextFieldRef
TextFieldExists( NSInteger tag ) = Boolean
```

```
Init
TextFieldInit( NSInteger tag, CGRect r ) = TextFieldRef// autoreleased
```

```
Selection and editing
TextFieldIsEditable( NSInteger tag ) = Boolean
TextFieldSetEditable( NSInteger tag, Boolean flag )
TextFieldIsSelectable( NSInteger tag ) = Boolean
TextFieldSetSelectable( NSInteger tag, Boolean flag )
```

```
Rich text behavior
TextFieldAllowsEditingTextAttributes( NSInteger tag ) = Boolean
TextFieldSetAllowsEditingTextAttributes( NSInteger tag, Boolean flag )
TextFieldImportsGraphics( NSInteger tag ) = Boolean
TextFieldSetImportsGraphics( NSInteger tag, Boolean flag )
```

```
Text color
TextFieldTextColor( NSInteger tag ) = ColorRef
TextFieldSetTextColor( NSInteger tag, ColorRef col )
```

```
Autolayout sizing
TextFieldPreferredMaxLayoutWidth( NSInteger tag ) = CGFloat// macOS 10.8+
TextFieldSetPreferredMaxLayoutWidth( NSInteger tag, CGFloat width )// macOS 10.8+
```

```
Background
TextFieldBackgroundColor( NSInteger tag ) = ColorRef
TextFieldSetBackgroundColor( NSInteger tag, ColorRef col )
TextFieldDrawsBackground( NSInteger tag ) = Boolean
TextFieldSetDrawsBackground( NSInteger tag, Boolean flag )
```

Border

```
TextFieldIsBezeled( NSInteger tag ) = Boolean
TextFieldSetBezeled( NSInteger tag, Boolean flag )
TextFieldBezelStyle( NSInteger tag ) = NSTextFieldBezelStyle
TextFieldSetBezelStyle( NSInteger tag, NSTextFieldBezelStyle style )
TextFieldIsBordered( NSInteger tag ) = Boolean
TextFieldSetBordered( NSInteger tag, Boolean flag )
```

#### Selecting text

```
TextFieldSelectText( NSInteger tag )
```

#### Responder

```
TextFieldAcceptsFirstResponder( NSInteger tag ) = Boolean
```

#### Instance properties

```
TextFieldAllowsDefaultTighteningForTruncation( NSInteger tag ) = Boolean// macOS 10.11+
TextFieldSetAllowsDefaultTighteningForTruncation( NSInteger tag, Boolean flag )// macOS 10.11+
TextFieldIsAutomaticTextCompletionEnabled( NSInteger tag ) = Boolean// macOS 10.12.2+
TextFieldSetAutomaticTextCompletionEnabled( NSInteger tag, Boolean flag )// macOS 10.12.2+
TextFieldMaximumNumberOfLines( NSInteger tag ) = NSInteger// macOS 10.11
TextFieldSetMaximumNumberOfLines( NSInteger tag, NSInteger lines )// macOS 10.11
TextFieldPlaceholderAttributedString( NSInteger tag ) = CFAttributedStringRef// macOS 10.10+
TextFieldSetPlaceholderAttributedString( NSInteger tag, CFAttributedStringRef string )// macOS 10.10
TextFieldPlaceholderString( NSInteger tag ) = CFStringRef// macOS 10.10+
TextFieldSetPlaceholderString( NSInteger tag, CFStringRef string )
```

#### Convenience

```
TextFieldSendsActionOnEndEditing( NSInteger tag ) = Boolean
TextFieldSetSendsActionOnEndEditing( NSInteger tag, Boolean flag )
TextFieldAllowsUndo( NSInteger tag ) = Boolean
TextFieldSetAllowsUndo( NSInteger tag, Boolean flag )
TextFieldSelectedRange( NSInteger tag ) = CFRange
TextFieldSetSelectedRange( NSInteger tag, CFRange range )
```

## See Also

[Control](#), [View](#)

## Apple documentation

[NSTextField](#)



---

### TextFinder

#### Functions

Convenience

```
TextFinderPerformAction( NSInteger tag, NSTextFinderAction action )
```

#### Apple documentation

[NSTextFinder](#)



TextLabel

statement

**Syntax**  
`TextLabel tag, text, rect, wndTag`

**Description**  
The **TextLabel** statement puts a new textlabel in the current output cocoa window, or alters an existing textlabel's characteristics.

Parameters	
<i>tag</i>	A number to identify the textlabel. A negative tag hides the textlabel.
<i>text</i>	The textlabel's text.
<i>rect</i>	Origin and size of the textlabel in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <code>CGRect</code> value
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

**Functions**  
See [TextField](#)

```
TextLabelWithTag( NSInteger tag ) = TextLabelRef
TextLabelExists( NSInteger tag ) = Boolean

Init
TextLabelInit( NSInteger tag, CGRect r ) = TextLabelRef// autoreleased
```

**See Also**  
[Control](#), [View](#)

**Apple documentation**  
[NSTextField](#)



---

## TextList

### Functions

Create

```
TextListWithMarkerFormat( CFStringRef format, NSUInteger options ) = TextListRef
```

Markers

```
TextListMarkerFormat( TextListRef ref ) = CFStringRef
```

```
TextListMarkerForItemNumber( TextListRef ref, NSUInteger itemNum ) = CFStringRef
```

Options

```
TextListOptions( TextListRef ref ) = NSUInteger
```

Item numbering

```
TextListStartingItemNumber( TextListRef ref ) = NSUInteger
```

### Apple documentation

[NSTextList](#)



Text ▶

TextMenu

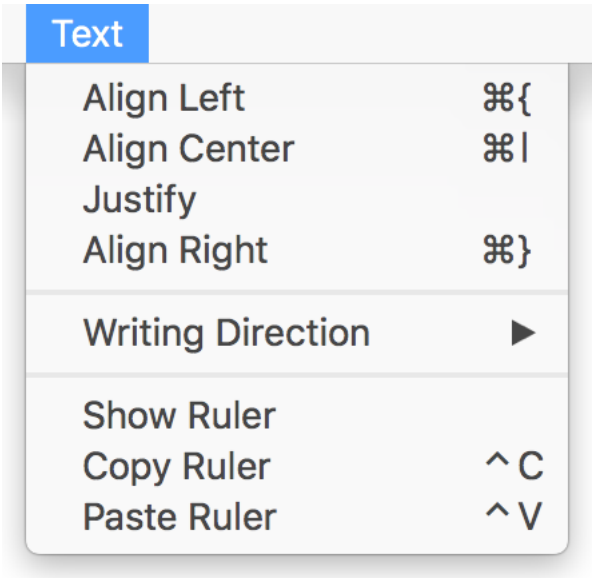
statement

Syntax

`textmenu menuIndex`

Description

Builds a default text menu. Requires the Cocoa runtime ([CocoaInit](#)).





---

### TextStorage

#### Functions

Layout managers

```
TextStorageLayoutManagers( TextStorageRef ref ) = CFArrayRef  
TextStorageAddLayoutManager( TextStorageRef ref, LayoutManagerRef lm )  
TextStorageRemoveLayoutManager( TextStorageRef ref, LayoutManagerRef lm )
```

Convenience

```
TextStorageWithURL( CFURLRef url, CFDictionaryRef options, CFDictionaryRef attributes, ErrorRef *err ) =  
TextStorageRef  
TextStorageSetWillProcessEditingCallback( TextStorageRef ref, ptr callback, ptr userData )  
TextStorageSetDidProcessEditingCallback( TextStorageRef ref, ptr callback, ptr userData )
```

#### Apple documentation

[NSTextStorage](#)



---

## TextTab

### Functions

Create

```
TextTabInit( NSTextAlignment alignment, CGFloat location, CFDictionaryRef options ) = TextTabRef// autoreleased
```

Tab stop info

```
TextTabLocation( TextTabRef ref ) = CGFloat
```

Text tab info

```
TextTabAlignment( TextTabRef ref ) = NSTextAlignment
```

```
TextTabOptions( TextTabRef ref ) = CFDictionaryRef
```

```
TextTabColumnTerminatorsForLocale( CFLocaleRef locale ) = CFCharacterSetRef// macOS 10.11+
```

### Apple documentation

[NSTextTab](#)





---

## TextTable

### Functions

Create

```
TextTableInit = TextTableRef// autoreleased
```

Columns

```
TextTableNumberOfColumns( TextTableRef ref ) = NSUInteger  
TextTableSetNumberOfColumns( TextTableRef ref, NSUInteger cols )
```

Layout algorithm

```
TextTableLayoutAlgorithm( TextTableRef ref ) = NSTextTableLayoutAlgorithm  
TextTableSetLayoutAlgorithm( TextTableRef ref, NSTextTableLayoutAlgorithm algorithm )
```

Collapsing borders

```
TextTableCollapsesBorders( TextTableRef ref ) = Boolean  
TextTableSetCollapsesBorders( TextTableRef ref, Boolean flag )
```

Hiding empty cells

```
TextTableHidesEmptyCells( TextTableRef ref ) = Boolean  
TextTableSetHidesEmptyCells( TextTableRef ref, Boolean flag )
```

Layout rectangles

```
TextTableRectForBlock( TextTableRef ref, TextTableBlockRef block, CGPoint startingPt, CGRect r, TextContainerRef  
textContainer, CFRange charRange ) = CGRect  
TextTableBoundsRectForBlock( TextTableRef ref, TextTableBlockRef block, CGRect contentRect, CGRect r,  
TextContainerRef textContainer, CFRange charRange ) = CGRect
```

Drawing

```
TextTableDrawBackgroundForBlock( TextTableRef ref, TextTableBlockRef block, CGRect frame, ViewRef controlView,  
CFRange charRange, LayoutManagerRef layoutManager )
```

### Apple documentation

[NSTextTable](#)



---

## TextTableBlock

### Functions

Create

```
TextTableBlockWithTable( TextTableRef table, NSInteger startingRow, NSInteger rowSpan, NSInteger startingColumn,
NSInteger columnSpan ) = TextTableBlockRef
```

Table

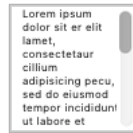
```
TextTableBlockTable( TextTableBlockRef ref ) = TextTableRef
```

Info

```
TextTableBlockStartingRow( TextTableBlockRef ref ) = NSInteger
TextTableBlockRowSpan( TextTableBlockRef ref ) = NSInteger
TextTableBlockStartingColumn( TextTableBlockRef ref ) = NSInteger
TextTableBlockColumnSpan( TextTableBlockRef ref ) = NSInteger
```

### Apple documentation

[NSTextTableBlock](#)



## TextView

statement

### Syntax

**textView** *tag*, *rect*, *scrollViewTag*, *textContainerRef*, *subclass*, *wndTag*

Note: The subclass parameter is deprecated. Use the [subclass](#) keyword instead.

### Description

The **textView** statement puts a new textview in the current output cocoa window, or alters an existing textview's characteristics.

### Parameters

<i>tag</i>	A number to identify the textview. A negative tag hides the textview.
<i>rect</i>	Origin and size of the textview in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>scrollViewTag</i>	The textView's scrollView tag (default = 0).
<i>textContainerRef</i>	Optional TextContainerRef.
<i>subclass</i>	A boolean value. When set to <code>_true</code> , the textview is subclassed and will send various events to the user on dialog function (see Dialog Events below).  Deprecated. Use the <a href="#">subclass</a> keyword instead.
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

`_textXxxx` - see [text](#)  
`_textViewDidChangeSelection`  
`_textViewWillShowSharingServicePicker` // macOS 10.8+  
`_textViewDoCommandBySelector`

### Functions

```
TextViewWithTag( NSInteger tag ) = TextViewRef  
TextViewExists( NSInteger tag ) = Boolean
```

```
Init  
TextViewInit( NSInteger tag, CGRect r ) = TextViewRef// autoreleased
```

Accessing text system objects

```
TextViewTextContainer( NSInteger tag ) = TextContainerRef  
TextViewSetTextContainer( NSInteger tag, TextContainerRef tc )  
TextViewReplaceTextContainer( NSInteger tag, TextContainerRef tc )  
TextViewTextContainerInset( NSInteger tag ) = CGSize  
TextViewSetTextContainerInset( NSInteger tag, CGSize inset )  
TextViewTextContainerOrigin( NSInteger tag ) = CGPoint  
TextViewInvalidateTextContainerOrigin( NSInteger tag )  
TextViewLayoutManager( NSInteger tag ) = LayoutManagerRef  
TextViewTextStorage( NSInteger tag ) = CFMutableAttributedString
```

Graphics attributes

```
TextViewBackgroundColor( NSInteger tag ) = ColorRef  
TextViewSetBackgroundColor( NSInteger tag, ColorRef col )  
TextViewDrawsBackground( NSInteger tag ) = Boolean  
TextViewSetDrawsBackground( NSInteger tag, Boolean flag )  
TextViewAllowsDocumentBackgroundColorChange( NSInteger tag ) = Boolean  
TextViewSetAllowsDocumentBackgroundColorChange( NSInteger tag, Boolean flag )  
TextViewChangeDocumentBackgroundColor( NSInteger tag )
```

Controlling display

```
TextViewShowFindIndicatorForRange( NSInteger tag, CFRange range )
```

Behavioral attributes

```
TextViewAllowsUndo( NSInteger tag ) = Boolean
```

```
TextViewSetAllowsUndo( NSInteger tag, Boolean flag )
TextViewIsEditable( NSInteger tag ) = Boolean
TextViewSetEditable( NSInteger tag, Boolean flag )
TextViewIsSelectable( NSInteger tag ) = Boolean
TextViewSetSelectable( NSInteger tag, Boolean flag )
TextViewIsRichText( NSInteger tag ) = Boolean
TextViewSetRichText( NSInteger tag, Boolean flag )
TextViewImportsGraphics( NSInteger tag ) = Boolean
TextViewSetImportsGraphics( NSInteger tag, Boolean flag )
TextViewSetBaseWritingDirection( NSInteger tag, NSWritingDirection direction )
TextViewDefaultParagraphStyle( NSInteger tag ) = ParagraphStyleRef
TextViewSetDefaultParagraphStyle( NSInteger tag, ParagraphStyleRef style )
TextViewOutline( NSInteger tag )
TextViewAllowsImageEditing( NSInteger tag ) = Boolean
TextViewSetAllowsImageEditing( NSInteger tag, Boolean flag )
TextViewIsAutomaticQuoteSubstitutionEnabled( NSInteger tag ) = Boolean
TextViewSetAutomaticQuoteSubstitutionEnabled( NSInteger tag, Boolean flag )
TextViewToggleAutomaticQuoteSubstitution( NSInteger tag )
TextViewIsAutomaticLinkDetectionEnabled( NSInteger tag ) = Boolean
TextViewSetAutomaticLinkDetectionEnabled( NSInteger tag, Boolean flag )
TextViewToggleAutomaticLinkDetection( NSInteger tag )
TextViewDisplaysLinkToolTips( NSInteger tag ) = Boolean
TextViewSetDisplaysLinkToolTips( NSInteger tag, Boolean flag )
```

#### Formatting controls

```
TextViewUsesRuler( NSInteger tag ) = Boolean
TextViewSetUsesRuler( NSInteger tag, Boolean flag )
TextViewIsRulerVisible( NSInteger tag ) = Boolean
TextViewSetRulerVisible( NSInteger tag, Boolean flag )
TextViewUsesInspectorBar( NSInteger tag ) = Boolean
TextViewSetUsesInspectorBar( NSInteger tag, Boolean flag )
```

#### Selection

```
TextViewSetSelectedRange( NSInteger tag, CFRange range )
TextViewSetSelectedRangeAffinity( NSInteger tag, CFRange range, NSSelectionAffinity affinity, Boolean stillSelecting )
TextViewSelectionAffinity( NSInteger tag ) = NSSelectionAffinity
TextViewSelectionGranularity( NSInteger tag ) = NSSelectionGranularity
TextViewSetSelectionGranularity( NSInteger tag, NSSelectionGranularity granularity )
TextViewInsertionPointColor( NSInteger tag ) = ColorRef
TextViewSetInsertionPointColor( NSInteger tag, ColorRef col )
TextViewSelectedTextAttributes( NSInteger tag ) = CFDictionaryRef
TextViewSetSelectedTextAttributes( NSInteger tag, CFDictionaryRef attributes )
TextViewMarkedTextAttributes( NSInteger tag ) = CFDictionaryRef
TextViewSetMarkedTextAttributes( NSInteger tag, CFDictionaryRef attributes )
TextViewLinkTextAttributes( NSInteger tag ) = CFDictionaryRef
TextViewSetLinkTextAttributes( NSInteger tag, CFDictionaryRef attributes )
TextViewCharacterIndexForInsertionAtPoint( NSInteger tag, CGPoint pt ) = NSUInteger
```

#### Managing the pasteboard

```
TextViewReadablePasteboardTypes( NSInteger tag ) = CFArrayRef
TextViewWritablePasteboardTypes( NSInteger tag ) = CFArrayRef
```

#### Text attributes

```
TextViewAlignJustified( NSInteger tag )
TextViewChangeAttributes( NSInteger tag )
TextViewChangeColor( NSInteger tag )
TextViewSetAlignment( NSInteger tag, NSTextAlignment alignment, CFRange range )
TextViewTypingAttributes( NSInteger tag ) = CFDictionaryRef
TextViewSetTypingAttributes( NSInteger tag, CFDictionaryRef attributes )
TextViewUseStandardKerning( NSInteger tag )
TextViewLowerBaseline( NSInteger tag )
TextViewRaiseBaseline( NSInteger tag )
TextViewTurnOffKerning( NSInteger tag )
TextViewLoosenKerning( NSInteger tag )
TextViewTightenKerning( NSInteger tag )
TextViewUseStandardLigatures( NSInteger tag )
TextViewTurnOffLigatures( NSInteger tag )
TextViewUseAllLigatures( NSInteger tag )
```

#### Clicking and pasting

```
TextViewPasteAsPlainText( NSInteger tag )
TextViewPasteAsRichText( NSInteger tag )
```

#### Undo support

```
TextViewBreakUndoCoalescing( NSInteger tag )
TextViewIsCoalescingUndo( NSInteger tag ) = Boolean
```

#### Subclasses to override

```
TextViewRangeForUserTextChange( NSInteger tag ) = CFRange
```

#### Spell checker

```
TextViewIsContinuousSpellCheckingEnabled( NSInteger tag ) = Boolean
TextViewSetContinuousSpellCheckingEnabled( NSInteger tag, Boolean flag )
TextViewToggleContinuousSpellChecking( NSInteger tag )
TextViewIsGrammarCheckingEnabled( NSInteger tag ) = Boolean
TextViewSetGrammarCheckingEnabled( NSInteger tag, Boolean flag )
TextViewToggleGrammarChecking( NSInteger tag )
TextViewSetSpellingState( NSInteger tag, NSInteger value, CFRange range )
```

#### Service picker

```
TextViewOrderFrontSharingServicePicker( NSInteger tag )// macOS 10.8+
```

#### Speaking

```
TextViewStartSpeaking( NSInteger tag )
TextViewStopSpeaking( NSInteger tag )
```

#### Working with panels

```
TextViewUsesFontPanel( NSInteger tag ) = Boolean
TextViewSetUsesFontPanel( NSInteger tag, Boolean flag )
TextViewUsesFindPanel( NSInteger tag ) = Boolean
TextViewSetUsesFindPanel( NSInteger tag, Boolean flag )
TextViewOrderFrontLinkPanel( NSInteger tag )
TextViewOrderFrontListPanel( NSInteger tag )
TextViewOrderFrontSpacingPanel( NSInteger tag )
TextViewOrderFrontTablePanel( NSInteger tag )
TextViewOrderFrontSubstitutionsPanel( NSInteger tag )
```

#### Changing layout orientation

```
TextViewChangeLayoutOrientation( NSInteger tag )
TextViewSetLayoutOrientation( NSInteger tag, NSTextLayoutOrientation orientation )
```

#### Find bar

```
TextViewUsesFindBar( NSInteger tag ) = Boolean
TextViewSetUsesFindBar( NSInteger tag, Boolean flag )
TextViewIsIncrementalSearchingEnabled( NSInteger tag ) = Boolean
TextViewSetIncrementalSearchingEnabled( NSInteger tag, Boolean flag )
```

#### Custom

```
TextViewPerformFindPanelAction( NSInteger tag, NSTextFinderAction action )
TextViewSetWordWrap( NSInteger tag, Boolean flag )
```

## See Also

[Text](#), [View](#)

## Apple documentation

[NSTextView](#)



---

## Thread

### Functions

Start

```
ThreadStart( ThreadRef thread )
```

Stop

```
ThreadSleepUntilDate( DateRef date )
```

```
ThreadSleepForTimeInterval( CFTimeInterval ti )
```

```
ThreadExit
```

```
ThreadCancel( ThreadRef thread )
```

Execution state

```
ThreadIsExecuting( ThreadRef thread ) = Boolean
```

```
ThreadIsFinished( ThreadRef thread ) = Boolean
```

```
ThreadIsCancelled( ThreadRef thread ) = Boolean
```

Main thread

```
ThreadIsMainThread( ThreadRef thread ) = Boolean// pass NULL for current thread
```

```
ThreadMainThread = ThreadRef
```

Query

```
ThreadIsMultiThreaded = Boolean
```

```
ThreadCurrentThread = ThreadRef
```

```
ThreadCallStackReturnAddresses = CFArrayRef
```

```
ThreadCallStackSymbols = CFArrayRef
```

Properties

```
ThreadDictionary( ThreadRef thread ) = CFMutableDictionaryRef
```

```
ThreadName( ThreadRef thread ) = CFStringRef
```

```
ThreadSetName( ThreadRef thread, CFStringRef name )
```

```
ThreadStackSize( ThreadRef thread ) = NSUInteger
```

Prioritizing

```
ThreadQualityOfService( ThreadRef thread ) = NSQualityOfService// macOS 10.10+
```

```
ThreadSetQualityOfService( ThreadRef thread, NSQualityOfService value )// macOS 10.10+
```

```
ThreadPriority( ThreadRef thread ) = double// pass NULL for current thread
```

```
ThreadSetPriority( ThreadRef thread, double priority )// pass NULL for current thread
```

Custom

```
ThreadDetachNewThreadFunction( ptr fnAddress, CFTypeRef object )
```

```
ThreadWithFunction( ptr fnAddress, CFTypeRef object ) = ThreadRef
```

### Apple documentation

[NSThread](#)



---

## Timer

### Functions

Create

```
TimerScheduledWithInterval( CTimeInterval ti, ptr callback, CTypeRef userInfo, Boolean repeats ) =  
CFRunLoopTimerRef  
TimerWithInterval( CTimeInterval ti, ptr callback, CTypeRef userInfo, Boolean repeats ) = CFRunLoopTimerRef  
TimerWithFireDate( CFDateRef dt, CTimeInterval ti, ptr callback, CTypeRef userInfo, Boolean repeats ) =  
CFRunLoopTimerRef
```

Fire

```
TimerFire( CFRunLoopTimerRef t )
```

Stop

```
TimerInvalidate( CFRunLoopTimerRef t )
```

Info

```
TimerIsValid( CFRunLoopTimerRef t ) = Boolean  
TimerFireDate( CFRunLoopTimerRef t ) = CFDateRef  
TimerTimeInterval( CFRunLoopTimerRef t ) = CTimeInterval  
TimerUserInfo( CFRunLoopTimerRef t ) = CTypeRef
```

Tolerance

```
TimerTolerance( CFRunLoopTimerRef t ) = CTimeInterval// macOS 10.9+
```

### Apple documentation

[NSTimer](#)



---

## TimeZone

### Functions

System time zones

```
TimeZoneLocal = CTimeZoneRef
TimeZoneSystem = CTimeZoneRef
TimeZoneResetSystem
TimeZoneDefault = CTimeZoneRef
```

Create

```
TimeZoneWithName( CFStringRef name ) = CTimeZoneRef
TimeZoneWithNameAndData( CFStringRef name, CFDataRef dta ) = CTimeZoneRef
TimeZoneWithAbbreviation( CFStringRef abbreviation ) = CTimeZoneRef
TimeZoneForSecondsFromGMT( NSInteger seconds ) = CTimeZoneRef
TimeZoneKnownNames = CFArrayRef
TimeZoneAbbreviationDictionary = CFDictionaryRef
```

Info

```
TimeZoneName( CTimeZoneRef ref ) = CFStringRef
TimeZoneAbbreviation( CTimeZoneRef ref ) = CFStringRef
TimeZoneAbbreviationForDate( CTimeZoneRef ref, CFDateRef dt ) = CFStringRef
TimeZoneSecondsFromGMT( CTimeZoneRef ref ) = NSInteger
TimeZoneSecondsFromGMTForDate( CTimeZoneRef ref, CFDateRef dt ) = NSInteger
TimeZoneData( CTimeZoneRef ref ) = CFDataRef
TimeZoneDataVersion = CFStringRef
```

Daylight savings

```
TimeZoneIsDaylightSavingTime( CTimeZoneRef ref ) = Boolean
TimeZoneIsDaylightSavingTimeForDate( CTimeZoneRef ref, CFDateRef dt ) = Boolean
TimeZoneDaylightSavingTimeOffset( CTimeZoneRef ref ) = CFTimeInterval
TimeZoneDaylightSavingTimeOffsetForDate( CTimeZoneRef ref, CFDateRef dt ) = CFTimeInterval
TimeZoneNextDaylightSavingTimeTransition( CTimeZoneRef ref ) = CFDateRef
TimeZoneNextDaylightSavingTimeTransitionAfterDate( CTimeZoneRef ref, CFDateRef dt ) = CFDateRef
```

Compare

```
TimeZoneIsEqual( CTimeZoneRef ref, CTimeZoneRef otherZone ) = Boolean
```

Describe

```
TimeZoneLocalizedName( CTimeZoneRef ref, NSTimeZoneNameStyle nameStyle, CFLocaleRef locale ) = CFStringRef
TimeZoneDescription( CTimeZoneRef ref ) = CFStringRef
```

### Apple documentation

[NSTimerZone](#)





TokenField

statement

Syntax

`tokenfield tag, enabled, text, rect, style, wndTag`

Description

The **tokenfield** statement puts a new tokenfield in the current output cocoa window, or alters an existing tokenfield's characteristics.

Parameters

<i>tag</i>	A number to identify the tokenfield. A negative tag hides the tokenfield.
<i>enabled</i>	Enables or disables the tokenfield.
<i>text</i>	The tokenfield's text.
<i>rect</i>	Origin and size of the tokenfield in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>style</i>	The tokenfield's token style.
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

Dialog Events

Event	Description
<code>_textFieldXxxx</code>	See <a href="#">textfield</a> .
<code>_tokenFieldStyleForRepresentedObject</code>	
<code>_tokenFieldCompletionsForSubstring</code>	
<code>_tokenFieldEditingStringForRepresentedObject</code>	
<code>_tokenFieldRepresentedObjectForEditingString</code>	
<code>_tokenFieldReadFromPasteboard</code>	
<code>_tokenFieldWriteRepresentedObjectToPasteboard</code>	
<code>_tokenFieldHasMenuForRepresentedObject</code>	
<code>_tokenFieldShouldAddObjects</code>	

Functions

`TokenFieldWithTag( NSInteger tag ) = TokenFieldRef`  
`TokenFieldExists( NSInteger tag ) = Boolean`

Init  
`TokenFieldInit( NSInteger tag, CGRect r ) = TokenFieldRef// autoreleased`

Style  
`TokenFieldStyle( NSInteger tag ) = NSUInteger`  
`TokenFieldSetStyle( NSInteger tag, NSUInteger style )`

Tokenizing character set  
`TokenFieldTokenizingCharacterSet( NSInteger tag ) = CFCharacterSetRef`  
`TokenFieldSetTokenizingCharacterSet( NSInteger tag, CFCharacterSetRef set )`  
`TokenFieldDefaultTokenizingCharacterSet = CFCharacterSetRef`

CompletionDelay  
`TokenFieldCompletionDelay( NSInteger tag ) = CFTimeInterval`  
`TokenFieldSetCompletionDelay( NSInteger tag, CFTimeInterval ti )`  
`TokenFieldDefaultCompletionDelay = CFTimeInterval`

fappCustom

`TextFieldRangesOfSelections( NSInteger tag ) = CFArrayRef` // returns an array of `CFRanges` as `ValueRefs`

## See Also

[Control](#), [View](#)

## Apple documentation

[NSTextField](#)



## Toolbar

statement

### Syntax

**toolbar** *tag*

### Description

Creates a new toolbar.

### Parameters

<i>tag</i>	A number to identify the toolbar. A negative tag hides the toolbar.
------------	---

### Functions

```
ToolbarWithTag( NSInteger tag ) = ToolbarRef
```

```
ToolbarExists( NSInteger tag ) = Boolean
```

#### Attributes

```
ToolbarDisplayMode( NSInteger tag ) = NSToolbarDisplayMode
ToolbarSetDisplayMode( NSInteger tag, NSToolbarDisplayMode mode )
ToolbarShowsBaselineSeparator( NSInteger tag ) = Boolean
ToolbarSetShowsBaselineSeparator( NSInteger tag, Boolean flag )
ToolbarAllowsUserCustomization( NSInteger tag ) = Boolean
ToolbarSetAllowsUserCustomization( NSInteger tag, Boolean flag )
ToolbarAllowsExtensionItems( NSInteger tag ) = Boolean// 10.10
ToolbarSetAllowsExtensionItems( NSInteger tag, Boolean flag )// 10.10
ToolbarSizeMode( NSInteger tag ) = NSToolbarSizeMode
ToolbarSetSizeMode( NSInteger tag, NSToolbarSizeMode size )
```

#### Display

```
ToolbarIsVisible( NSInteger tag ) = Boolean
ToolbarSetVisible( NSInteger tag, Boolean flag )
```

#### Customization

```
ToolbarRunCustomizationPalette( NSInteger tag )
ToolbarCustomizationPaletteIsRunning( NSInteger tag ) = Boolean
```

#### Autosaving the configuration

```
ToolbarAutosavesConfiguration( NSInteger tag ) = Boolean
ToolbarSetAutosavesConfiguration( NSInteger tag, Boolean flag )
ToolbarConfigurationDictionary( NSInteger tag ) = CFDictionaryRef
ToolbarSetConfigurationFromDictionary( NSInteger tag, CFDictionaryRef dict )
```

### See Also

[ToolbarItem](#), [NibToolbar](#)

### Apple documentation

[NSToolbar](#)



## ToolbarItem

statement

### Syntax

```
toolbaritem tag, identifier, label, image, allowed, default, toolbarTag
```

### Description

Adds a new toolbar item to the current toolbar in the current output cocoa window. Toolbaritems can only be added to an existing toolbar. Toolbaritems are added from left to right.

### Parameters

<i>tag</i>	A number to identify the toolbaritem.
<i>identifier</i>	The kind of toolbar item (i.e. separator, space, font, print, custom etc.).
<i>label</i>	Text to identify the item when the toolbar is displaying text descriptions.
<i>image</i>	This param can be the name of (or path to) an image resource, or an ImageRef (NSImage).
<i>allowed</i>	Boolean controls whether toolbaritem is shown in the "allowed" set
<i>default</i>	Boolean controls whether toolbaritem is shown in the "default" set
<i>toolbarTag</i>	The tag value of a previously created toolbar.

### Dialog Events

Event	Description
<i>_toolbarItemClick</i>	The user clicked a toolbaritem.
<i>_toolbarItemValidate</i>	Validate the toolbaritem. Call DialogEventSetBool( <i>_true/_false</i> ) to enable/disable the item.

### Functions

```
ToolbarItemWithTag( NSInteger tag ) = ToolbarItemRef  
ToolbarItemExists( NSInteger toolbarTag, NSInteger itemTag ) = Boolean
```

#### Attributes

```
ToolbarItemIdentifier( NSInteger toolbarTag, NSInteger itemTag ) = CFStringRef  
ToolbarItemLabel( NSInteger toolbarTag, NSInteger itemTag ) = CFStringRef  
ToolbarItemPaletteLabel( NSInteger toolbarTag, NSInteger itemTag ) = CFStringRef  
ToolbarItemSetPaletteLabel( NSInteger toolbarTag, NSInteger itemTag, CFStringRef label )  
ToolbarItemToolTip( NSInteger toolbarTag, NSInteger itemTag ) = CFStringRef  
ToolbarItemSetToolTip( NSInteger toolbarTag, NSInteger itemTag, CFStringRef toolTip )  
ToolbarItemMenuFormRepresentation( NSInteger toolbarTag, NSInteger itemTag ) = CocoaMenuRef  
ToolbarItemSetMenuFormRepresentation( NSInteger toolbarTag, NSInteger itemTag, CocoaMenuRef menu )  
ToolbarItemIsEnabled( NSInteger toolbarTag, NSInteger itemTag ) = Boolean  
ToolbarItemSetEnabled( NSInteger toolbarTag, NSInteger itemTag, Boolean flag )  
ToolbarItemImage( NSInteger toolbarTag, NSInteger itemTag ) = ImageRef  
ToolbarItemView( NSInteger toolbarTag, NSInteger itemTag ) = ViewRef  
ToolbarItemSetView( NSInteger toolbarTag, NSInteger itemTag, NSInteger vwTag )  
ToolbarItemMinSize( NSInteger toolbarTag, NSInteger itemTag ) = CGSize  
ToolbarItemSetMinSize( NSInteger toolbarTag, NSInteger itemTag, CGSize size )  
ToolbarItemMaxSize( NSInteger toolbarTag, NSInteger itemTag ) = CGSize  
ToolbarItemSetMaxSize( NSInteger toolbarTag, NSInteger itemTag, CGSize size )
```

#### VisibilityPriority

```
ToolbarItemVisibilityPriority( NSInteger toolbarTag, NSInteger itemTag ) = NSInteger  
ToolbarItemSetVisibilityPriority( NSInteger toolbarTag, NSInteger itemTag, NSInteger priority )
```

#### Validation

```
ToolbarItemValidate( NSInteger toolbarTag, NSInteger itemTag )  
ToolbarItemAutovalidates( NSInteger toolbarTag, NSInteger itemTag ) = Boolean  
ToolbarItemSetAutovalidates( NSInteger toolbarTag, NSInteger itemTag, Boolean flag )
```

#### Controlling duplicates

```
ToolbarItemAllowsDuplicatesInToolbar( NSInteger toolbarTag, NSInteger itemTag ) = Boolean
```

## See Also

[Toolbar](#)

## Apple documentation

[NSToolbarItem](#)



---

## Touch

### Functions

Type

```
TouchType( TouchRef ref ) = NSTouchType// macOS 10.12.2+
```

Properties of this touch

```
TouchIdentity( TouchRef ref ) = CTypeRef
```

```
TouchPhase( TouchRef ref ) = NSTouchPhase
```

```
TouchNormalizedPosition( TouchRef ref ) = CGPoint
```

```
TouchIsResting( TouchRef ref ) = Boolean// macOS 10.10+
```

Properties of touch device

```
TouchDevice( TouchRef ref ) = CTypeRef
```

```
TouchDeviceSize( TouchRef ref ) = CGSize
```

Touch location

```
TouchLocationInView( TouchRef ref, NSInteger tag ) = CGPoint// macOS 10.12.2+
```

```
TouchPreviousLocationInView( TouchRef ref, NSInteger tag ) = CGPoint// macOS 10.12.2+
```

### Apple documentation

[NSTouch](#)



---

## UndoManager

### Functions

#### Registering

```
UndoManagerRegisterUndo( UndoManagerRef ref, ptr fnAddress, CTypeRef object )
```

#### Ability

```
UndoManagerCanUndo( UndoManagerRef ref ) = Boolean
```

```
UndoManagerCanRedo( UndoManagerRef ref ) = Boolean
```

#### Perform

```
UndoManagerUndo( UndoManagerRef ref )
```

```
UndoManagerUndoNestedGroup( UndoManagerRef ref )
```

```
UndoManagerRedo( UndoManagerRef ref )
```

#### Limiting

```
UndoManagerLevelsOfUndo( UndoManagerRef ref ) = NSUInteger
```

```
UndoManagerSetLevelsOfUndo( UndoManagerRef ref, NSUInteger levels )
```

#### Groups

```
UndoManagerBeginUndoGrouping( UndoManagerRef ref )
```

```
UndoManagerEndUndoGrouping( UndoManagerRef ref )
```

```
UndoManagerGroupsByEvent( UndoManagerRef ref ) = Boolean
```

```
UndoManagerSetGroupsByEvent( UndoManagerRef ref, Boolean flag )
```

```
UndoManagerGroupingLevel( UndoManagerRef ref ) = NSInteger
```

#### Enable

```
UndoManagerDisableUndoRegistration( UndoManagerRef ref )
```

```
UndoManagerEnableUndoRegistration( UndoManagerRef ref )
```

```
UndoManagerIsUndoRegistrationEnabled( UndoManagerRef ref ) = Boolean
```

#### Performed

```
UndoManagerIsUndoing( UndoManagerRef ref ) = Boolean
```

```
UndoManagerIsRedoing( UndoManagerRef ref ) = Boolean
```

#### Clearing

```
UndoManagerRemoveAllActions( UndoManagerRef ref )
```

#### Action name

```
UndoManagerUndoActionName( UndoManagerRef ref ) = CFStringRef
```

```
UndoManagerRedoActionName( UndoManagerRef ref ) = CFStringRef
```

```
UndoManagerSetActionName( UndoManagerRef ref, CFStringRef name )
```

#### Menu item title

```
UndoManagerUndoMenuItemTitle( UndoManagerRef ref ) = CFStringRef
```

```
UndoManagerRedoMenuItemTitle( UndoManagerRef ref ) = CFStringRef
```

```
UndoManagerUndoMenuTitleForUndoActionName( UndoManagerRef ref, CFStringRef name ) = CFStringRef
```

```
UndoManagerRedoMenuTitleForUndoActionName( UndoManagerRef ref, CFStringRef name ) = CFStringRef
```

#### Loops

```
UndoManagerRunLoopModes( UndoManagerRef ref ) = CFArrayRef
```

#### Discardable

```
UndoManagerSetActionIsDiscardable( UndoManagerRef ref, Boolean flag )
```

```
UndoManagerUndoActionIsDiscardable( UndoManagerRef ref ) = Boolean
```

```
UndoManagerRedoActionIsDiscardable( UndoManagerRef ref ) = Boolean
```

### Apple documentation

[NSUndoManager](#)



## URL

---

### Functions

#### Create

```
URLWithString( CFStringRef string ) = CFURLRef
URLWithStringRelativeToURL( CFStringRef string, CFURLRef baseURL ) = CFURLRef
URLFileURLWithPath( CFStringRef string ) = CFURLRef
URLFileURLWithPathComponents( CFArrayRef components ) = CFURLRef
URLByResolvingBookmarkData( CFDataRef bookmarkData, NSURLBookmarkResolutionOptions options, CFURLRef relativeURL,
Boolean *isStale, ErrorRef *err ) = CFURLRef
```

#### Compare

```
URLIsEqual( CFURLRef url1, CFURLRef url2 ) = Boolean
```

#### Accessing parts

```
URLPath( CFURLRef url ) = CFStringRef
URLPathComponents( CFURLRef url ) = CFArrayRef
URLLastPathComponent( CFURLRef url ) = CFStringRef
URLPathExtension( CFURLRef url ) = CFStringRef
```

#### Accessing resource values

```
URLResourceValuesForKeys( CFURLRef url, CFArrayRef keys ) = CFDictionaryRef
URLGetResourceValueForKey( CFURLRef url, ptr key, CTypeRef *value, ErrorRef *err ) = Boolean
URLSetResourceValueForKey( CFURLRef url, ptr key, CTypeRef value, ErrorRef *err ) = Boolean
URLSetResourceValues( CFURLRef url, CFDictionaryRef keyedValues, ErrorRef *err ) = Boolean
```

#### Modifying and converting

```
URLByAppendingPathComponent( CFURLRef url, CFStringRef pathComponent ) = CFURLRef
URLByAppendingPathExtension( CFURLRef url, CFStringRef extension ) = CFURLRef
URLByDeletingLastPathComponent( CFURLRef url ) = CFURLRef
URLByDeletingPathExtension( CFURLRef url ) = CFURLRef
```

#### Bookmark data

```
URLBookmarkDataWithContentsOfURL( CFURLRef url, ErrorRef *err ) = CFDataRef
URLBookmarkDataWithOptions( CFURLRef url, NSURLBookmarkCreationOptions options, CFArrayRef resourceKeys, CFURLRef
relativeURL, ErrorRef *err ) = CFDataRef
URLResourceValuesForKeysFromBookmarkData( CFArrayRef resourceKeys, CFDataRef bookmarkData ) = CFDictionaryRef
URLWriteBookmarkData( CFDataRef bookmarkData, CFURLRef toURL, NSURLBookmarkFileCreationOptions options, ErrorRef
*err ) = Boolean
```

#### Pasteboards

```
URLFromPasteboard( CocoaPasteboardRef pb ) = CFURLRef
URLWriteToPasteboard( CFURLRef url, CocoaPasteboardRef pb )
```

### Apple documentation

[NSURL](#)





---

## URLAsset

### Functions

Create

```
URLAssetWithURL( CFURLRef url, CFDictionaryRef options ) = URLAssetRef
```

URL

```
URLAssetURL( URLAssetRef ref ) = CFURLRef
```

Supported media types

```
URLAssetAudiovisualTypes = CFArrayRef
```

### Apple documentation

[AVURLAsset](#)



## NSURLSession

macOS 10.9+

### Functions

Create

```
NSURLSessionWithConfiguration( NSURLSessionConfigurationRef config ) = NSURLSessionRef
NSURLSessionWithDelegateQueue( NSURLSessionConfigurationRef config, OperationQueueRef queue ) = NSURLSessionRef
NSURLSessionSharedSession = NSURLSessionRef
```

Configure

```
NSURLSessionConfiguration( NSURLSessionRef ref ) = NSURLSessionConfigurationRef
NSURLSessionDelegateQueue( NSURLSessionRef ref ) = OperationQueueRef
NSURLSessionDescription( NSURLSessionRef ref ) = CFStringRef
```

Add data tasks

```
NSURLSessionDataTaskWithURL( NSURLSessionRef ref, CFURLRef url ) = NSURLSessionDataTaskRef
```

```
NSURLSessionDataTaskWithURLCompletionHandler( NSURLSessionRef ref, CFURLRef url, ptr handlerAddress, ptr userData ) =
NSURLSessionDataTaskRef
```

Completion handler params:

```
( session as NSURLSessionRef, dta as CFDataRef, response as URLResponseRef, err as ErrorRef, userData as ptr )
```

```
NSURLSessionDataTaskWithRequest( NSURLSessionRef ref, URLRequestRef request ) = NSURLSessionDataTaskRef
```

```
NSURLSessionDataTaskWithRequestCompletionHandler( NSURLSessionRef ref, URLRequestRef request, ptr handlerAddress, ptr
userData ) = NSURLSessionDataTaskRef
```

Completion handler params:

```
( session as NSURLSessionRef, dta as CFDataRef, response as URLResponseRef, err as ErrorRef, userData as ptr )
```

Add download tasks

```
NSURLSessionDownloadTaskWithURL( NSURLSessionRef ref, CFURLRef url ) = NSURLSessionDownloadTaskRef
```

```
NSURLSessionDownloadTaskWithURLCompletionHandler( NSURLSessionRef ref, CFURLRef url, ptr handlerAddress, ptr userData ) =
NSURLSessionDownloadTaskRef
```

Completion handler params:

```
( session as NSURLSessionRef, url as CFURLRef, response as URLResponseRef, err as ErrorRef, userData as ptr )
```

```
NSURLSessionDownloadTaskWithRequest( NSURLSessionRef ref, URLRequestRef request ) = NSURLSessionDownloadTaskRef
```

```
NSURLSessionDownloadTaskWithRequestCompletionHandler( NSURLSessionRef ref, URLRequestRef request, ptr handlerAddress,
ptr userData ) = NSURLSessionDownloadTaskRef
```

Completion handler params:

```
( session as NSURLSessionRef, url as CFURLRef, response as URLResponseRef, err as ErrorRef, userData as ptr )
```

```
NSURLSessionDownloadTaskWithResumeData( NSURLSessionRef ref, CFDataRef dta ) = NSURLSessionDownloadTaskRef
```

```
NSURLSessionDownloadTaskWithResumeDataCompletionHandler( NSURLSessionRef ref, CFDataRef dta, ptr handlerAddress, ptr
userData ) = NSURLSessionDownloadTaskRef
```

Completion handler params:

```
( session as NSURLSessionRef, url as CFURLRef, response as URLResponseRef, err as ErrorRef, userData as ptr )
```

Add upload tasks

```
NSURLSessionUploadTaskWithRequestFromData( NSURLSessionRef ref, URLRequestRef request, CFDataRef dta ) =
NSURLSessionUploadTaskRef
```

```
NSURLSessionUploadTaskWithRequestFromDataCompletionHandler( NSURLSessionRef ref, URLRequestRef request, CFDataRef dta,
ptr handlerAddress, ptr userData ) = NSURLSessionUploadTaskRef
```

Completion handler params:

```
( session as NSURLSessionRef, dta as CFDataRef, response as URLResponseRef, err as ErrorRef, userData as ptr )
```

```
NSURLSessionUploadTaskWithRequestFromFile( NSURLSessionRef ref, URLRequestRef request, CFURLRef url ) =
NSURLSessionUploadTaskRef
```

```
NSURLSessionUploadTaskWithRequestFromFileCompletionHandler( NSURLSessionRef ref, URLRequestRef request, CFURLRef url,
ptr handlerAddress, ptr userData ) = NSURLSessionUploadTaskRef
```

Completion handle params:

```
( session as NSURLSessionRef, dta as CFDataRef, response as URLResponseRef, err as ErrorRef, userData as ptr )
```

```
NSURLSessionUploadTaskWithStreamedRequest( NSURLSessionRef ref, URLRequestRef request ) = NSURLSessionUploadTaskRef
```

Add stream tasks

```
NSURLSessionStreamTaskWithHostName( NSURLSessionRef ref, CFStringRef hostName, NSInteger port ) =
NSURLSessionStreamTaskRef// macOS 10.11+
```

```
NSURLSessionStreamTaskWithNetService( NSURLSessionRef ref, NetServiceRef service ) = NSURLSessionStreamTaskRef// macOS
10.11+
```

Managing session

```
NSURLSessionFinishTasksAndInvalidate( NSURLSessionRef ref )
NSURLSessionFlushWithCompletionHandler( NSURLSessionRef ref, ptr handlerAddress, ptr userData )
    Completion handler params:
    ( session as NSURLSessionRef, userData as ptr )

NSURLSessionGetTasksWithCompletionHandler( NSURLSessionRef ref, ptr handlerAddress, ptr userData )
    Completion handler params:
    ( session as NSURLSessionRef, dataTasks as CFArrayRef, uploadTasks as CFArrayRef, downloadTasks as CFArrayRef,
    userData as ptr )

NSURLSessionGetAllTasksWithCompletionHandler( NSURLSessionRef ref, ptr handlerAddress, ptr userData )// macOS 10.11+
NSURLSessionInvalidateAndCancel( NSURLSessionRef ref )
    Completion handler function params:
    ( session as NSURLSessionRef, tasks as CFArrayRef, userData as ptr )

NSURLSessionResetWithCompletionHandler( NSURLSessionRef ref, ptr handlerAddress, ptr userData )
    Completion handler params:
    ( session as NSURLSessionRef, userData as ptr )
```

## Apple documentation

[NSURLSession](#)



## NSURLSessionConfiguration

macOS 10.9+

### Functions

Create

```
NSURLSessionConfigurationDefault = NSURLSessionConfigurationRef
NSURLSessionConfigurationEphemeral = NSURLSessionConfigurationRef
NSURLSessionConfigurationBackground( CFStringRef identifier ) = NSURLSessionConfigurationRef// macOS 10.10+
```

General properties

```
NSURLSessionConfigurationIdentifier( NSURLSessionConfigurationRef ref ) = CFStringRef
NSURLSessionConfigurationHTTPAdditionalHeaders( NSURLSessionConfigurationRef ref ) = CFDictionaryRef
NSURLSessionConfigurationSetHTTPAdditionalHeaders( NSURLSessionConfigurationRef ref, CFDictionaryRef headers )
NSURLSessionConfigurationNetworkServiceType( NSURLSessionConfigurationRef ref ) = NSURLRequestNetworkServiceType
NSURLSessionConfigurationSetNetworkServiceType( NSURLSessionConfigurationRef ref, NSURLRequestNetworkServiceType type )
NSURLSessionConfigurationAllowsCellularAccess( NSURLSessionConfigurationRef ref ) = Boolean
NSURLSessionConfigurationSetAllowsCellularAccess( NSURLSessionConfigurationRef ref, Boolean flag )
NSURLSessionConfigurationTimeoutIntervalForRequest( NSURLSessionConfigurationRef ref ) = CFTimeInterval
NSURLSessionConfigurationSetTimeoutIntervalForRequest( NSURLSessionConfigurationRef ref, CFTimeInterval ti )
NSURLSessionConfigurationTimeoutIntervalForResource( NSURLSessionConfigurationRef ref ) = CFTimeInterval
NSURLSessionConfigurationSetTimeoutIntervalForResource( NSURLSessionConfigurationRef ref, CFTimeInterval ti )
NSURLSessionConfigurationSharedContainerIdentifier( NSURLSessionConfigurationRef ref ) = CFStringRef// macOS 10.10+
NSURLSessionConfigurationSetSharedContainerIdentifier( NSURLSessionConfigurationRef ref, CFStringRef identifier )//
macOS 10.10+
NSURLSessionConfigurationWaitsForConnectivity( NSURLSessionConfigurationRef ref ) = Boolean// macOS 10.13+
NSURLSessionConfigurationSetWaitsForConnectivity( NSURLSessionConfigurationRef ref, Boolean flag )// macOS 10.13+
```

Cookie policies

```
NSURLSessionConfigurationHTTPCookieAcceptPolicy( NSURLSessionConfigurationRef ref ) = NSHTTPCookieAcceptPolicy
NSURLSessionConfigurationSetHTTPCookieAcceptPolicy( NSURLSessionConfigurationRef ref, NSHTTPCookieAcceptPolicy policy )
NSURLSessionConfigurationHTTPShouldSetCookies( NSURLSessionConfigurationRef ref ) = Boolean
NSURLSessionConfigurationSetHTTPShouldSetCookies( NSURLSessionConfigurationRef ref, Boolean flag )
NSURLSessionConfigurationHTTPCookieStorage( NSURLSessionConfigurationRef ref ) = HTTPCookieStorageRef
NSURLSessionConfigurationSetHTTPCookieStorage( NSURLSessionConfigurationRef ref, HTTPCookieStorageRef storage )
```

Security policies

```
NSURLSessionConfigurationTLSMaximumSupportedProtocol( NSURLSessionConfigurationRef ref ) = SSLProtocol
NSURLSessionConfigurationSetTLSMaximumSupportedProtocol( NSURLSessionConfigurationRef ref, SSLProtocol protocol )
NSURLSessionConfigurationTLSMinimumSupportedProtocol( NSURLSessionConfigurationRef ref ) = SSLProtocol
NSURLSessionConfigurationSetTLSMinimumSupportedProtocol( NSURLSessionConfigurationRef ref, SSLProtocol protocol )
NSURLSessionConfigurationURLCredentialStorage( NSURLSessionConfigurationRef ref ) = URLCredentialStorageRef
NSURLSessionConfigurationSetURLCredentialStorage( NSURLSessionConfigurationRef ref, URLCredentialStorageRef storage )
```

Caching policies

```
NSURLSessionConfigurationURLCache( NSURLSessionConfigurationRef ref ) = URLCacheRef
NSURLSessionConfigurationSetURLCache( NSURLSessionConfigurationRef ref, URLCacheRef cache )
NSURLSessionConfigurationRequestCachePolicy( NSURLSessionConfigurationRef ref ) = NSURLRequestCachePolicy
NSURLSessionConfigurationSetRequestCachePolicy( NSURLSessionConfigurationRef ref, NSURLRequestCachePolicy policy )
```

Background transfers

```
NSURLSessionConfigurationIsDiscretionary( NSURLSessionConfigurationRef ref ) = Boolean// macOS 10.10+
NSURLSessionConfigurationSetDiscretionary( NSURLSessionConfigurationRef ref, Boolean flag )// macOS 10.10+
NSURLSessionConfigurationShouldUseExtendedBackgroundIdleMode( NSURLSessionConfigurationRef ref ) = Boolean// macOS
10.11+
NSURLSessionConfigurationSetShouldUseExtendedBackgroundIdleMode( NSURLSessionConfigurationRef ref, Boolean flag )//
macOS 10.11+
```

Custom protocols

```
NSURLSessionConfigurationProtocoClasses( NSURLSessionConfigurationRef ref ) = CFArrayRef
NSURLSessionConfigurationSetProtocoClasses( NSURLSessionConfigurationRef ref, CFArrayRef classes )
```

HTTP policy and proxy properties

```
NSURLSessionConfigurationHTTPMaximumConnectionsPerHost( NSURLSessionConfigurationRef ref ) = NSInteger
NSURLSessionConfigurationSetHTTPMaximumConnectionsPerHost( NSURLSessionConfigurationRef ref, NSInteger connections )
NSURLSessionConfigurationHTTPShouldUsePipelining( NSURLSessionConfigurationRef ref ) = Boolean
NSURLSessionConfigurationSetHTTPShouldUsePipelining( NSURLSessionConfigurationRef ref, Boolean flag )
NSURLSessionConfigurationConnectionProxyDictionary( NSURLSessionConfigurationRef ref ) = CFDictionaryRef
NSURLSessionConfigurationSetConnectionProxyDictionary( NSURLSessionConfigurationRef ref, CFDictionaryRef proxy )
```

### Apple documentation

[NSURLSessionConfiguration](#)



---

### NSURLSessionDownloadTask

macOS 10.9+

#### Functions

Cancel download

```
NSURLSessionDownloadTaskCancelByProducingResumeData( NSURLSessionDownloadTaskRef ref, ptr handlerAddress, ptr userData )
    Handler function params:
    ( task as NSURLSessionDownloadTaskRef, resumeData as CFDataRef, userData as ptr )
```

#### Apple documentation

[NSURLSessionDownloadTask](#)



---

## NSURLSessionStreamTask

macOS 10.11+

### Functions

Reading and writing

```
NSURLSessionStreamTaskReadData( NSURLSessionStreamTaskRef ref, NSUInteger minLength, NSUInteger maxLength,
CFTimeInterval timeout, ptr handlerAddress, ptr userData )
```

Handler function params:

```
( task as NSURLSessionStreamTaskRef, dta as CFDataRef, atEOF as Boolean, err as ErrorRef, userData as ptr )
```

```
NSURLSessionStreamTaskWriteData( NSURLSessionStreamTaskRef ref, CFDataRef dta, CFTimeInterval timeout, ptr
handlerAddress, ptr userData )
```

Handler function params:

```
( task as NSURLSessionStreamTaskRef, err as ErrorRef, userData as ptr )
```

Capture streams

```
NSURLSessionStreamCaptureStreams( NSURLSessionStreamTaskRef ref )
```

```
NSURLSessionStreamCloseWrite( NSURLSessionStreamTaskRef ref )
```

Secure connections

```
NSURLSessionStreamStartSecureConnection( NSURLSessionStreamTaskRef ref )
```

```
NSURLSessionStreamStopSecureConnection( NSURLSessionStreamTaskRef ref )
```

### Apple documentation

[NSURLSessionStreamTask](#)



---

## NSURLSessionTask

macOS 10.9+

### Functions

Task state

```
NSURLSessionTaskCancel( NSURLSessionTaskRef ref )
NSURLSessionTaskResume( NSURLSessionTaskRef ref )
NSURLSessionTaskSuspend( NSURLSessionTaskRef ref )
NSURLSessionTaskState( NSURLSessionTaskRef ref ) = NSURLSessionTaskState
NSURLSessionTaskPriority( NSURLSessionTaskRef ref ) = float// macOS 10.10+
NSURLSessionTaskSetPriority( NSURLSessionTaskRef ref, float priority )// macOS 10.10+
```

Task progress

```
NSURLSessionTaskProgress( NSURLSessionTaskRef ref ) = ProgressRef// macOS 10.13+
NSURLSessionTaskCountOfBytesExpectedToReceive( NSURLSessionTaskRef ref ) = SInt64
NSURLSessionTaskCountOfBytesReceived( NSURLSessionTaskRef ref ) = SInt64
NSURLSessionTaskCountOfBytesExpectedToSend( NSURLSessionTaskRef ref ) = SInt64
NSURLSessionTaskCountOfBytesSent( NSURLSessionTaskRef ref ) = SInt64
```

General task info

```
NSURLSessionTaskCurrentRequest( NSURLSessionTaskRef ref ) = URLRequestRef
NSURLSessionTaskOriginalRequest( NSURLSessionTaskRef ref ) = URLRequestRef
NSURLSessionTaskResponse( NSURLSessionTaskRef ref ) = URLResponseRef
NSURLSessionTaskDescription( NSURLSessionTaskRef ref ) = CFStringRef
NSURLSessionTaskSetDescription( NSURLSessionTaskRef ref, CFStringRef description )
NSURLSessionTaskIdentifier( NSURLSessionTaskRef ref ) = NSUInteger
NSURLSessionTaskError( NSURLSessionTaskRef ref ) = ErrorRef
```

Scheduling tasks

```
NSURLSessionTaskCountOfBytesClientExpectsToReceive( NSURLSessionTaskRef ref ) = SInt64// macOS 10.13+
NSURLSessionTaskCountOfBytesClientExpectsToSend( NSURLSessionTaskRef ref ) = SInt64// macOS 10.13+
NSURLSessionTaskEarliestBeginDate( NSURLSessionTaskRef ref ) = CFDateRef// macOS 10.13+
```

### Apple documentation

[NSURLSessionTask](#)



---

## UserDefaults

### Functions

#### Get

```
UserDefaultsObject( CFStringRef key ) = CTypeRef
UserDefaultsBool( CFStringRef key ) = Boolean
UserDefaultsInteger( CFStringRef key ) = long
UserDefaultsDouble( CFStringRef key ) = double
UserDefaultsFloat( CFStringRef key ) = float
UserDefaultsURL( CFStringRef key ) = CFURLRef
UserDefaultsArray( CFStringRef key ) = CFArrayRef
UserDefaultsDictionary( CFStringRef key ) = CFDictionaryRef
UserDefaultsString( CFStringRef key ) = CFStringRef
UserDefaultsData( CFStringRef key ) = CFDataRef
UserDefaultsDictionaryRepresentation = CFDictionaryRef
```

#### Set

```
UserDefaultsSetObject( CFStringRef key, CTypeRef obj )
UserDefaultsSetBool( CFStringRef key, Boolean value )
UserDefaultsSetDouble( CFStringRef key, double value )
UserDefaultsSetInteger( CFStringRef key, long value )
UserDefaultsSetFloat( CFStringRef key, float value )
UserDefaultsSetURL( CFStringRef key, CFURLRef value )
```

#### Remove

```
UserDefaultsRemoveObject( CFStringRef key )
```

#### Suites

```
UserDefaultsAddSuiteNamed( CFStringRef name )
UserDefaultsRemoveSuiteNamed( CFStringRef name )
```

#### Register

```
UserDefaultsRegisterDefaults( CFDictionaryRef defaults )
```

#### Legacy

```
UserDefaultsSynchronize = Boolean
```

#### Convenience

```
UserDefaultsColor( CFStringRef key ) = ColorRef
UserDefaultsSetColor( CFStringRef key, ColorRef col )
UserDefaultsFont( CFStringRef key ) = CTNSFontRef
UserDefaultsSetFont( CFStringRef key, CTNSFontRef font )
UserDefaultsDictationMenuItemSetDisabled( Boolean flag )
UserDefaultsCharacterPaletteMenuItemSetDisabled( Boolean flag )
```

#### Custom

```
UserDefaultsStoreWindowViewValues( NSInteger wndTag, CFStringRef key )
UserDefaultsRestoreWindowViewValues( NSInteger wndTag, CFStringRef key )
UserDefaultsRemoveWindowViewValues( NSInteger wndTag, CFStringRef key )
```

### Apple documentation

[NSUserDefaults](#)





---

### UserDefaultsController

#### Functions

Shared instance

```
UserDefaultsControllerShared = UserDefaultsControllerRef
```

Values

```
UserDefaultsControllerDefaults = UserDefaultsRef  
UserDefaultsControllerInitialValues = CFDictionaryRef  
UserDefaultsControllerSetInitialValues( CFDictionaryRef values )  
UserDefaultsControllerHasUnappliedChanges = Boolean  
UserDefaultsControllerAppliesImmediately = Boolean  
UserDefaultsControllerSetAppliesImmediately( Boolean flag )  
UserDefaultsControllerValues = CFTypeRef  
UserDefaultsControllerRevert  
UserDefaultsControllerRevertToInitialValues  
UserDefaultsControllerSave
```

Convenience

```
UserDefaultsControllerValuesSetValueForKey( CFTypeRef value, CFStringRef key )
```

#### Apple documentation

[NSUserDefaultsController](#)



## UserNotification

### Functions

Init  
`UserNotificationInit = UserNotificationRef` // autoreleased

Display info  
`UserNotificationTitle( UserNotificationRef ref ) = CFStringRef`  
`UserNotificationSetTitle( UserNotificationRef ref, CFStringRef title )`  
`UserNotificationSubtitle( UserNotificationRef ref ) = CFStringRef`  
`UserNotificationSetSubtitle( UserNotificationRef ref, CFStringRef subtitle )`  
`UserNotificationInformativeText( UserNotificationRef ref ) = CFStringRef`  
`UserNotificationSetInformativeText( UserNotificationRef ref, CFStringRef infoText )`  
`UserNotificationContentImage( UserNotificationRef ref ) = ImageRef` // macOS 10.9+  
`UserNotificationSetContentImage( UserNotificationRef ref, ImageRef image )` // macOS 10.9+  
`UserNotificationIdentifier( UserNotificationRef ref ) = CFStringRef` // macOS 10.9+  
`UserNotificationSetIdentifier( UserNotificationRef ref, CFStringRef identifier )` // macOS 10.9+  
`UserNotificationResponse( UserNotificationRef ref ) = CFAttributedStringRef` // macOS 10.9+  
`UserNotificationResponsePlaceholder( UserNotificationRef ref ) = CFStringRef` // macOS 10.9+

Buttons  
`UserNotificationHasActionButton( UserNotificationRef ref ) = Boolean`  
`UserNotificationSetHasActionButton( UserNotificationRef ref, Boolean flag )`  
`UserNotificationActionButtonTitle( UserNotificationRef ref ) = CFStringRef`  
`UserNotificationSetActionButtonTitle( UserNotificationRef ref, CFStringRef title )`  
`UserNotificationOtherButtonTitle( UserNotificationRef ref ) = CFStringRef`  
`UserNotificationSetOtherButtonTitle( UserNotificationRef ref, CFStringRef title )`  
`UserNotificationHasReplyButton( UserNotificationRef ref ) = Boolean` // macOS 10.9+  
`UserNotificationSetHasReplyButton( UserNotificationRef ref, Boolean flag )` // macOS 10.9+

Timing  
`UserNotificationDeliveryDate( UserNotificationRef ref ) = CFDateRef`  
`UserNotificationSetDeliveryDate( UserNotificationRef ref, CFDateRef dt )`  
`UserNotificationActualDeliveryDate( UserNotificationRef ref ) = CFDateRef`  
`UserNotificationDeliveryRepeatInterval( UserNotificationRef ref ) = DateComponentsRef`  
`UserNotificationSetDeliveryRepeatInterval( UserNotificationRef ref, DateComponentsRef components )`  
`UserNotificationDeliveryTimeZone( UserNotificationRef ref ) = CFTimeZoneRef`  
`UserNotificationSetDeliveryTimeZone( UserNotificationRef ref, CFTimeZoneRef zone )`

Delivery info  
`UserNotificationIsPresented( UserNotificationRef ref ) = Boolean`  
`UserNotificationIsRemote( UserNotificationRef ref ) = Boolean`  
`UserNotificationSoundName( UserNotificationRef ref ) = CFStringRef`  
`UserNotificationSetSoundName( UserNotificationRef ref, CFStringRef name )`

Activation method  
`UserNotificationActivationType( UserNotificationRef ref ) = NSUserNotificationActivationType`  
`UserNotificationAdditionalActivationAction( UserNotificationRef ref ) = UserNotificationActionRef` // macOS 10.10+  
`UserNotificationAdditionalActions( UserNotificationRef ref ) = CFArrayRef` // macOS 10.10+  
`UserNotificationSetAdditionalActions( UserNotificationRef ref, CFArrayRef actions )` // macOS 10.10+

User info  
`UserNotificationUserInfo( UserNotificationRef ref ) = CFDictionaryRef`  
`UserNotificationSetUserInfo( UserNotificationRef ref, CFDictionaryRef userInfo )`

Convenience  
`UserNotificationWithTitle( CFStringRef title, CFStringRef subtitle, CFStringRef infoText, CFStringRef sndName ) = UserNotificationRef` // autoreleased

### Apple documentation

[NSUserNotification](#)



---

### UserNotificationAction

#### Functions

Create

```
UserNotificationActionInit( CFStringRef identifier, CFStringRef title ) = UserNotificationActionRef// autoreleased
```

Identifier and title

```
UserNotificationActionIdentifier( UserNotificationActionRef ref ) = CFStringRef
```

```
UserNotificationActionTitle( UserNotificationActionRef ref ) = CFStringRef
```

#### Apple documentation

[NSUserNotificationAction](#)



---

## UserNotificationCenter

### Functions

Default center

```
UserNotificationCenterDefault = UserNotificationCenterRef
```

Queue

```
UserNotificationCenterScheduleNotification( UserNotificationRef ref )
```

```
UserNotificationCenterScheduledNotifications = CFArrayRef
```

```
UserNotificationCenterRemoveScheduledNotification( UserNotificationRef ref )
```

Deliver

```
UserNotificationCenterDeliverNotification( UserNotificationRef ref )
```

```
UserNotificationCenterDeliveredNotifications = CFArrayRef
```

```
UserNotificationCenterRemoveDeliveredNotification( UserNotificationRef ref )
```

```
UserNotificationCenterRemoveAllDeliveredNotifications
```

Convenience

```
UserNotificationCenterScheduleNotificationWithTitle( CFStringRef title, CFStringRef subtitle, CFStringRef infoText, CFStringRef sndName )
```

```
UserNotificationCenterDeliverNotificationWithTitle( CFStringRef title, CFStringRef subtitle, CFStringRef infoText, CFStringRef sndName )
```

### Apple documentation

[NSUserNotificationCenter](#)



---

### va\_list

#### Functions

```
va_start( va_list ap, ptr lastArg )  
va_copy( va_list ap2, va_list ap1 )  
va_end( va_list ap )
```

```
va_argInt32( va_list ap ) = SInt32  
va_argInt64( va_list ap ) = SInt64  
va_argLong( va_list ap ) = long  
va_argDouble( va_list ap ) = double  
va_argObj( va_list ap ) = CTypeRef
```



---

## Value

### Functions

Pointer and object values

```
ValueWithPointer( ptr p ) = ValueRef
ValuePointer( ValueRef v ) = ptr
ValueWithRange( CFRange range ) = ValueRef
ValueRange( ValueRef v ) = CFRange
ValueWithPoint( CGPoint pt ) = ValueRef
ValuePoint( ValueRef v ) = CGPoint
ValueWithSize( CGSize size ) = ValueRef
ValueSize( ValueRef v ) = CGSize
ValueWithRect( CGRect r ) = ValueRef
ValueRect( ValueRef v ) = CGRect
```

CATransform3DAdditions

```
ValueWithCATransform3D( CATransform3D t ) = ValueRef
```

### Apple documentation

[NSValue](#)



## View

statement

### Syntax

```
view tag, rect, subclass, wndTag
```

### Description

The **view** statement puts a new view in the current output cocoa window, or alters an existing view's characteristics.

Note: The subclass parameter is deprecated. Use the [subclass](#) keyword instead.

### Parameters

<i>tag</i>	A number to identify the view. A negative tag hides the view.
<i>rect</i>	Origin and size of the view in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>subclass</i>	A boolean value. When set to <code>_true</code> , the view is subclassed and will send various events to the user on dialog function (see dialog events below).  Deprecated. Use the <a href="#">subclass</a> keyword instead.
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

Window content views (plus views with their 'subclass' parameter set to `_true`) trigger these events.

Event	Description
<code>_viewDrawRect</code>	DialogEventRect returns the view's dirtyRect.
<code>_viewKey</code>	The user has pressed a key.
<code>_viewKeyUp</code>	The user has released a key.
<code>_viewFlagsChanged</code>	Sent when a modifier key has been pressed or released.
<code>_viewMouseDown</code>	The user has pressed the left mouse button.
<code>_viewMouseDragged</code>	The user has moved the mouse with the left button pressed.
<code>_viewMouseUp</code>	The user has released the left mouse button.
<code>_viewMouseMoved</code>	The mouse has moved.
<code>_viewMouseEntered</code>	The cursor has entered the view's tracking rectangle.
<code>_viewMouseExited</code>	The cursor has exited the view's tracking rectangle.
<code>_viewRightMouseDown</code>	The user has pressed the right mouse button.
<code>_viewRightMouseDragged</code>	The user has moved the mouse with the right button pressed.
<code>_viewRightMouseUp</code>	The user has released the right mouse button.

### Functions

```
ViewWithTag( NSInteger tag ) = ViewRef
```

```
ViewExists( NSInteger tag ) = Boolean
```

```
Init  
ViewInit( NSInteger tag, CGRect r ) = ViewRef// autoreleased
```

### Hierarchy

```
ViewSuperview( NSInteger tag ) = NSInteger
```

```
ViewSubviews( NSInteger tag ) = CFArrayRef returns an array of ViewRefs
```

```
ViewWindow( NSInteger tag ) = NSInteger
```

```
ViewAddSubview( NSInteger viewTag, NSInteger subviewTag )
```

```
ViewAddSubviewPositioned( NSInteger superviewTag, NSInteger subviewTag, UIWindowOrderingMode position, NSInteger otherViewTag )
ViewRemoveFromSuperview( NSInteger tag )
ViewIsDescendantOf( NSInteger tag, NSInteger superTag ) = Boolean
ViewOpaqueAncestor( NSInteger tag ) = NSInteger
ViewAncestorSharedWithView( NSInteger tag, NSInteger otherTag ) = NSInteger
ViewSortSubviewsUsingFunction( NSInteger tag, ptr fnAddress, ptr context )
```

#### Frame

```
ViewFrame( NSInteger tag ) = CGRect
ViewSetFrame( NSInteger tag, CGRect r )
ViewSetFrameOrigin( NSInteger tag, CGPoint origin )
ViewSetFrameSize( NSInteger tag, CGSize size )
ViewFrameRotation( NSInteger tag ) = CGFloat
ViewSetFrameRotation( NSInteger tag, CGFloat rotation )
```

#### Bounds

```
ViewBounds( NSInteger tag ) = CGRect
ViewSetBounds( NSInteger tag, CGRect bounds )
ViewSetBoundsOrigin( NSInteger tag, CGPoint )
ViewSetBoundsSize( NSInteger tag, CGSize size )
ViewBoundsRotation( NSInteger tag ) = CGFloat
ViewSetBoundsRotation( NSInteger tag, CGFloat rotation )
```

#### Layer

```
ViewSetWantsLayer( NSInteger tag, Boolean flag )
ViewLayer( NSInteger tag ) = CALayerRef
ViewSetLayer( NSInteger tag, CALayerRef layer )
```

#### Layer-related properties

```
ViewAlphaValue( NSInteger tag ) = CGFloat
ViewSetAlphaValue( NSInteger tag, CGFloat value )
ViewFrameCenterRotation( NSInteger tag ) = CGFloat
ViewSetFrameCenterRotation( NSInteger tag, CGFloat rotation )
ViewShadow( NSInteger tag ) = ShadowRef
ViewSetShadow( NSInteger tag, ShadowRef shadow )
```

#### Drawing

```
ViewVisibleRect( NSInteger tag ) = CGRect
```

#### Printing

```
ViewPrint( NSInteger tag )
```

#### Invalidate view content

```
ViewSetNeedsDisplay( NSInteger tag )
ViewSetNeedsDisplayInRect( NSInteger tag, CGRect r )
ViewIsOpaque( NSInteger tag ) = Boolean
```

#### Converting coordinate values

```
ViewConvertPointFromView( NSInteger tag1, CGPoint pt, NSInteger tag2 ) = CGPoint
```

#### Modify coordinate system

```
ViewRotateByAngle( NSInteger tag, CGFloat angle )
```

#### Examine coordinate system modifications

```
ViewIsFlipped( NSInteger tag ) = Boolean
ViewSetFlipped( NSInteger tag, Boolean flag )
```

#### Resizing subviews

```
ViewAutoresizesSubviews( NSInteger tag ) = Boolean
ViewSetAutoresizesSubviews( NSInteger tag, Boolean flag )
ViewAutoresizingMask( NSInteger tag ) = NSAutoresizingMaskOptions
ViewSetAutoresizingMask( NSInteger tag, NSAutoresizingMaskOptions mask )
```

#### Layout anchors

```
ViewBottomAnchor( NSInteger tag ) = LayoutYAxisAnchorRef// macOS 10.11+
ViewCenterXAnchor( NSInteger tag ) = LayoutXAxisAnchorRef// macOS 10.11+
ViewCenterYAnchor( NSInteger tag ) = LayoutYAxisAnchorRef// macOS 10.11+
ViewFirstBaselineAnchor( NSInteger tag ) = LayoutYAxisAnchorRef// macOS 10.11+
ViewHeightAnchor( NSInteger tag ) = LayoutDimensionRef// macOS 10.11+
ViewLastBaselineAnchor( NSInteger tag ) = LayoutYAxisAnchorRef// macOS 10.11+
ViewLeadingAnchor( NSInteger tag ) = LayoutXAxisAnchorRef// macOS 10.11+
ViewLeftAnchor( NSInteger tag ) = LayoutXAxisAnchorRef// macOS 10.11+
ViewRightAnchor( NSInteger tag ) = LayoutXAxisAnchorRef// macOS 10.11+
ViewTopAnchor( NSInteger tag ) = LayoutYAxisAnchorRef// macOS 10.11+
ViewTrailingAnchor( NSInteger tag ) = LayoutXAxisAnchorRef// macOS 10.11+
ViewWidthAnchor( NSInteger tag ) = LayoutDimensionRef// macOS 10.11+
```

#### Constraints

```
ViewConstraints( NSInteger tag ) = CFArrayRef
ViewAddConstraint( NSInteger tag, LayoutConstraintRef constraint )
```



```
ViewAddConstraints( NSInteger tag, NSArrayRef constraints )
ViewRemoveConstraint( NSInteger tag, LayoutConstraintRef constraint )
ViewRemoveConstraints( NSInteger tag, NSArrayRef constraints )
```

#### Layout guides

```
ViewAddLayoutGuide( NSInteger tag, LayoutGuideRef guide )// macOS 10.11+
ViewLayoutGuides( NSInteger tag ) = NSArrayRef// macOS 10.11+
ViewRemoveLayoutGuide( NSInteger tag, LayoutGuideRef guide )// macOS 10.11+
```

#### Measuring in auto layout

```
ViewFittingSize( NSInteger tag ) = CGSize
ViewIntrinsicContentSize( NSInteger tag ) = CGSize
ViewInvalidateIntrinsicContentSize( NSInteger tag )
```

#### Focusing

```
ViewLockFocus( NSInteger tag )
ViewUnlockFocus( NSInteger tag )
ViewFocusView = NSInteger
```

#### Vibrancy

```
ViewAllowsVibrancy( NSInteger tag ) = Boolean// macOS 10.10+
```

#### Hiding

```
ViewIsHidden( NSInteger tag ) = Boolean
ViewSetHidden( NSInteger tag, Boolean flag )
ViewIsHiddenOrHasHiddenAncestor( NSInteger tag ) = Boolean
```

#### Gesture recognizer

```
ViewGestureRecognizers( NSInteger tag ) = NSArrayRef
ViewAddGestureRecognizer( NSInteger tag, GestureRecognizerRef ref )
ViewRemoveGestureRecognizer( NSInteger tag, GestureRecognizerRef ref )
```

#### Event handling

```
ViewHitTest( NSInteger tag, CGPoint pt ) = NSInteger
```

#### Key-view loop management

```
ViewNextKeyView( NSInteger tag ) = NSInteger
ViewSetNextKeyView( NSInteger tag, NSInteger nextViewTag )
ViewNextValidKeyView( NSInteger tag ) = NSInteger
ViewPreviousKeyView( NSInteger tag ) = NSInteger
ViewPreviousValidKeyView( NSInteger tag ) = NSInteger
```

#### Scrolling

```
ViewScrollRect( NSInteger tag, CGRect r, CGSize size )
ViewEnclosingScrollView( NSInteger tag ) = NSInteger
```

#### Dragging

```
ViewRegisterForDraggedTypes( NSInteger tag, NSArrayRef types )
```

#### Notifications

```
ViewPostsFrameChangedNotifications( NSInteger tag ) = Boolean
ViewSetPostsFrameChangedNotifications( NSInteger tag, Boolean flag )
ViewPostsBoundsChangedNotifications( NSInteger tag ) = Boolean
ViewSetPostsBoundsChangedNotifications( NSInteger tag, Boolean flag )
```

#### Searching by tag

```
ViewWithTag( NSInteger tag ) = ptr
ViewTag( ViewRef view ) = NSInteger
```

#### Tooltips

```
ViewRemoveAllToolTips( NSInteger tag )
ViewSetToolTip( NSInteger tag, CFStringRef string )
```

#### Tracking rect

```
ViewAddTrackingRect( NSInteger tag, CGRect r, ObjectRef owner, ptr userData, Boolean assumeInside ) =
NSTrackingRectTag
ViewRemoveTrackingRect( NSInteger tag, NSTrackingRectTag rectTag )
```

#### Tracking area

```
ViewAddTrackingArea( NSInteger tag, TrackingAreaRef ta )
ViewRemoveTrackingArea( NSInteger tag, TrackingAreaRef ta )
ViewTrackingAreas( NSInteger tag ) = NSArrayRef
```

#### Cursor tracking

```
ViewAddCursorRect( NSInteger tag, CGRect r, CursorRef curs )
ViewRemoveCursorRect( NSInteger tag, CGRect r, CursorRef curs )
ViewDiscardCursorRects( NSInteger tag )
```

#### Definition windows

```
ViewShowDefinition( NSInteger tag, CFAttributedStringRef aString, CGPoint pt )
```

```
ViewShowDefinitionBaselineOriginProvider( NSInteger tag, CFAttributedStringRef aString, CFRange targetRange,
CFDictionaryRef options, ptr originProvider, ptr userData )
```

Appearance customization

```
ViewAppearance( NSInteger tag ) = AppearanceRef// macOS 10.9+
ViewEffectiveAppearance( NSInteger tag ) = AppearanceRef// macOS 10.9+
```

Convenience

```
ViewRegisterForDraggedFileTypes( NSInteger tag, CFTyperef fileTypes )
    fileTypes is an array or semicolon delimited string of types.
ViewRegisterForDraggedFileExtensions( NSInteger tag, CFTyperef extensions )
    extensions is an array or semicolon delimited string of extensions.
ViewDragInside( NSInteger tag ) = Boolean
ViewSetDragInside( NSInteger tag, Boolean flag )
```

```
ViewsEmbedInView( NSInteger tag, ... ) // a comma-separated list of widget tags, ending with NULL
```

```
ViewSetAcceptsFirstResponder( NSInteger tag, Boolean flag )
```

```
ViewAnimator( NSInteger tag ) = ptr
ViewAnimatorSetFrame( NSInteger tag, CGRect r )
ViewAnimatorSetFrameOrigin( NSInteger tag, CGPoint origin )
ViewAnimatorSetFrameSize( NSInteger tag, CGSize size )
ViewAnimatorSetFrameRotation( NSInteger tag, CGFloat rot )
ViewAnimatorSetBounds( NSInteger tag, CGRect r )
ViewAnimatorSetBoundsOrigin( NSInteger tag, CGPoint origin )
ViewAnimatorSetBoundsSize( NSInteger tag, CGSize size )
ViewAnimatorSetBoundsRotation( NSInteger tag, CGFloat rot )
ViewAnimatorSetFrameCenterRotation( NSInteger tag, CGFloat rot )
ViewAnimatorSetAlphaValue( NSInteger tag, CGFloat value )
```

```
ViewSetClickGestureRecognizerCallback( NSInteger tag, NSInteger numberOfClicks, ptr callback, ptr userData )// macOS
10.10+
ViewSetMagnificationGestureRecognizerCallback( NSInteger tag, ptr callback, ptr userData )// macOS 10.10+
ViewSetPanGestureRecognizerCallback( NSInteger tag, ptr callback, ptr userData )// macOS 10.10+
ViewSetPressGestureRecognizerCallback( NSInteger tag, ptr callback, ptr userData )// macOS 10.10+
ViewSetRotationGestureRecognizerCallback( NSInteger tag, ptr callback, ptr userData )// macOS 10.10+
```

Appearance

```
ViewSetAppearance( NSInteger tag, AppearanceRef appearance )
ViewSetAppearanceNamed( NSInteger tag, CFStringRef name )
```

Most appropriate appearance

```
AppearanceBestMatchFromAppearancesWithNames( AppearanceRef ref, CFArrayRef names ) = CFStringRef// macOS 10.14+
```

Custom

```
ViewProperty( NSInteger tag, CFStringRef key ) = CFTyperef
ViewSetProperty( NSInteger tag, CFStringRef key, CFTyperef value )
ViewRemoveProperty( NSInteger tag, CFStringRef key )
ViewRemoveAllProperties( NSInteger tag )
ViewSuspendDrawRect( NSInteger tag, Boolean flag )
ViewDrawRectIsSuspended( NSInteger tag ) = Boolean
```

## Apple documentation

[NSView](#)



---

## ViewController

### Functions

Init

```
ViewControllerInit = ViewControllerRef// autoreleased
```

View properties

```
ViewControllerView( ViewControllerRef controller ) = NSInteger
```

```
ViewControllerSetView( ViewControllerRef controller, NSInteger vwTag )
```

### Apple documentation

[NSViewController](#)



ViewMenu

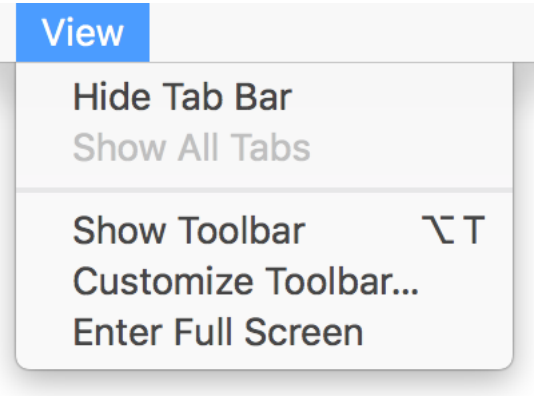
statement

Syntax

`viewmenu menuIndex`

Description

Builds a default view menu. Requires the Cocoa runtime ([CocoaInit](#)).





## VisualEffectView

statement

### Syntax

```
visualeffectview tag, rect, material, mode, state, image, emphasized, wndTag
```

### Description

The **visualeffectview** statement puts a new visualeffectview in the current output window, or alters an existing visualeffectview's characteristics.

### Parameters

<i>tag</i>	A number to identify the view. A negative tag hides the view.
<i>rect</i>	Origin and size of the view in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>material</i>	See VisualEffectView.incl header for options.
<i>mode</i>	See VisualEffectView.incl header for options.
<i>state</i>	See VisualEffectView.incl header for options.
<i>image</i>	This param can be NULL or one of the following: 1. The name of (or path to) an image resource. 2. An ImageRef (NSImage).
<i>emphasized</i>	A boolean value to indicate if the view is emphasized. Requires macOS 10.12+.
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Functions

```
VisualEffectViewWithTag( NSInteger tag ) = VisualEffectViewRef
```

```
VisualEffectViewExists( NSInteger tag ) = Boolean
```

Init

```
VisualEffectViewInit( NSInteger tag, CGRect r ) = VisualEffectViewRef// autoreleased
```

Background material

```
VisualEffectViewMaterial( NSInteger tag ) = NSVisualEffectMaterial
```

```
VisualEffectViewSetMaterial( NSInteger tag, NSVisualEffectMaterial material )
```

Appearance

```
VisualEffectViewBlendingMode( NSInteger tag ) = NSVisualEffectBlendingMode
```

```
VisualEffectViewSetBlendingMode( NSInteger tag, NSVisualEffectBlendingMode mode )
```

```
VisualEffectViewIsEmphasized( NSInteger tag ) = Boolean// macOS 10.12+
```

```
VisualEffectViewSetEmphasized( NSInteger tag, Boolean flag )// macOS 10.12+
```

```
VisualEffectViewInteriorBackgroundStyle( NSInteger tag ) = NSBackgroundStyle
```

Masking

```
VisualEffectViewMaskImage( NSInteger tag ) = ImageRef
```

```
VisualEffectViewSetMaskImage( NSInteger tag, ImageRef image )
```

Enable/disable

```
VisualEffectViewState( NSInteger tag ) = NSVisualEffectState
```

```
VisualEffectViewSetState( NSInteger tag, NSVisualEffectState state )
```

### Apple documentation

[NSVisualEffectView](#)



## WKWebView

statement

### Syntax

**wkwebview** *tag, rect, configuration, wndTag*

### Description

The **wkwebview** statement puts a new wkwebview in the current output cocoa window, or alters an existing wkwebview characteristics.

### Parameters

<i>tag</i>	A number to identify the webview. A negative tag hides the view.
<i>rect</i>	Origin and size of the view in window coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <code>CGRect</code> value
<i>configuration</i>	A <code>WKWebViewConfiguration</code> object (optional).
<i>wndTag</i>	Optional parameter for when the target window is not the current output window. Note: specifying this parameter does not bring the window forward or make it the output window.

### Dialog Events

```
_webViewDidCommitNavigation
_webViewDidStartProvisionalNavigation
_webViewDidReceiveServerRedirectForProvisionalNavigation
_webViewDidReceiveAuthenticationChallenge
_webViewDidFailNavigation
_webViewDidFailProvisionalNavigation
_webViewDidFinishNavigation
_webViewWebContentProcessDidTerminate
_webViewDecidePolicyForNavigationAction
```

### Functions

```
WKWebViewWithTag( NSInteger tag ) = WKWebViewRef
WKWebViewExists( NSInteger tag ) = Boolean
```

Can load content

```
WKWebViewHandlesURLScheme( CFStringRef scheme ) = Boolean// macOS 10.13+
```

Init

```
wkWebViewConfiguration( NSInteger tag ) = WKWebViewConfigurationRef// lowercase 'wk' to avoid clash with class name
```

Info

```
WKWebViewTitle( NSInteger tag ) = CFStringRef
WKWebViewURL( NSInteger tag ) = CFURLRef
WKWebViewCustomUserAgent( NSInteger tag ) = CFStringRef// macOS 10.11+
```

Loading content

```
WKWebViewEstimatedProgress( NSInteger tag ) = double
WKWebViewHasOnlySecureContent( NSInteger tag ) = Boolean
WKWebViewLoadHTMLString( NSInteger tag, CFStringRef string, CFURLRef baseURL ) = WKNavigationRef
WKWebViewIsLoading( NSInteger tag ) = Boolean
WKWebViewReload( NSInteger tag ) = WKNavigationRef
WKWebViewReloadFromOrigin( NSInteger tag ) = WKNavigationRef
WKWebViewStopLoading( NSInteger tag )
WKWebViewLoadData( NSInteger tag, CFDataRef dta, CFStringRef mimeType, CFStringRef charEncodingName, CFURLRef
baseURL ) = WKNavigationRef// macOS 10.11+
WKWebViewLoadFileURL( NSInteger tag, CFURLRef fileURL, CFURLRef readAccessURL ) = WKNavigationRef// macOS 10.11+
```

Scaling content

```
WKWebViewAllowsMagnification( NSInteger tag ) = Boolean
WKWebViewSetAllowsMagnification( NSInteger tag, Boolean flag )
WKWebViewMagnification( NSInteger tag ) = CGFloat
WKWebViewSetMagnification( NSInteger tag, CGFloat magnification )
WKWebViewSetMagnificationCenteredAtPoint( NSInteger tag, CGFloat magnification, CGPoint pt )
```

Navigating

```
WKWebViewAllowsBackForwardNavigationGestures( NSInteger tag ) = Boolean
WKWebViewSetAllowsBackForwardNavigationGestures( NSInteger tag, Boolean flag )
```

```
WKWebViewBackForwardList( NSInteger tag ) = WKBackForwardListRef
WKWebViewCanGoBack( NSInteger tag ) = Boolean
WKWebViewCanGoForward( NSInteger tag ) = Boolean
WKWebViewAllowsLinkPreview( NSInteger tag ) = Boolean// macOS 10.11+
WKWebViewSetAllowsLinkPreview( NSInteger tag, Boolean flag )// macOS 10.11+
WKWebViewGoBack( NSInteger tag ) = WKNavigationRef
WKWebViewGoForward( NSInteger tag ) = WKNavigationRef
WKWebViewGoToBackForwardListItem( NSInteger tag, WKBackForwardListItemRef itemRef ) = WKNavigationRef
WKWebViewLoadRequest( NSInteger tag, URLRequestRef request ) = WKNavigationRef
```

Apple documentation

[WKWebView](#)



## Cocoa Window

statement

### Syntax

**cocoa window** *tag, title, rect, style*

### Description

Use this statement to:

- Create a new cocoa window;
- Activate and bring to the front an existing cocoa window;
- Make an existing cocoa window visible or invisible;
- Alter the title or rectangle of an existing cocoa window.

### Parameters

<i>tag</i>	Unique number to identify the window. A negative value hides the window.
<i>title</i>	The window title.
<i>rect</i>	Origin and size of the window's content area in screen coordinates. This can be specified in either of two ways: (1) (x,y)-(w,h) or (x,y,w,h) (2) <a href="#">CGRect</a> value
<i>style</i>	The window's style can be any of the following options. They can be combined using '+'.  NSBorderlessWindowMask, NSTitledWindowMask, NSClosableWindowMask, NSMiniaturizableWindowMask, NSResizableWindowMask, NSTexturedBackgroundWindowMask, NSUnifiedTitleAndToolbarWindowMask, NSFulScreenWindowMask, NSFulSizeContentViewWindowMask, NSUtilityWindowMask, NSDocModalWindowMask, NSNonactivatingPanelMask, NSHUDWindowMask  Default: NSTitledWindowMask + NSClosableWindowMask + NSMiniaturizableWindowMask + NSResizableWindowMask

### Window Content View

\_windowContentViewTag - all window content views are given this tag value. Substituting this constant in many of the ViewXxxx functions manipulates the front (key) window's content view.

### Dialog Events

[\\_windowWillBeginSheet](#)  
[\\_windowDidEndSheet](#)  
[\\_windowWillResize](#)  
[\\_windowDidResize](#)  
[\\_windowWillStartLiveResize](#)  
[\\_windowDidEndLiveResize](#)  
[\\_windowWillMiniaturize](#)  
[\\_windowDidMiniaturize](#)  
[\\_windowDidDeminiaturize](#)  
[\\_windowShouldZoom](#)  
[\\_windowWillEnterFullScreen](#)  
[\\_windowDidEnterFullScreen](#)  
[\\_windowWillExitFullScreen](#)  
[\\_windowDidExitFullScreen](#)  
[\\_windowDidFailToEnterFullScreen](#)  
[\\_windowDidFailToExitFullScreen](#)  
[\\_windowWillMove](#)  
[\\_windowDidMove](#)  
[\\_windowDidChangeScreen](#)  
[\\_windowShouldClose](#)  
[\\_windowWillClose](#)  
[\\_windowDidBecomeKey](#)  
[\\_windowDidResignKey](#)  
[\\_windowDidBecomeMain](#)  
[\\_windowDidResignMain](#)  
[\\_windowDidUpdate](#)

Dragging events

[\\_windowDraggingEntered](#)  
[\\_windowWantsPeriodicDraggingUpdates](#)  
[\\_windowDraggingUpdated](#)



```
_windowDraggingEnded
_windowDraggingExited
_windowPrepareForDragOperation
_windowPerformDragOperation
_windowConcludeDragOperation
_windowUpdateDraggingItemsForDrag
```

#### Subclass events

```
_windowMouseDown
_windowMouseUp
_windowKeyDown
_windowKeyUp
_windowFlagsChanged
_windowMouseEntered
_windowMouseExited
```

## Functions

```
WindowWithTag( NSInteger tag ) = CocoaWindowRef
WindowExists( NSInteger tag ) = Boolean
```

#### Configure

```
WindowToggleFullScreen( NSInteger tag )
WindowWorksWhenModal( NSInteger tag ) = Boolean
WindowAlphaValue( NSInteger tag ) = CGFloat
WindowSetAlphaValue( NSInteger tag, CGFloat value )
WindowBackgroundColor( NSInteger tag ) = ColorRef
WindowSetBackgroundColor( NSInteger tag, ColorRef col )
WindowColorSpace( NSInteger tag ) = ColorSpaceRef
WindowSetColorSpace( NSInteger tag, ColorSpaceRef cs )
WindowCanHide( NSInteger tag ) = Boolean
WindowSetCanHide( NSInteger tag, Boolean flag )
WindowIsOnActiveSpace( NSInteger tag ) = Boolean
WindowHidesOnDeactivate( NSInteger tag ) = Boolean
WindowSetHidesOnDeactivate( NSInteger tag, Boolean flag )
WindowCollectionBehavior( NSInteger tag ) = NSWindowCollectionBehavior
WindowSetCollectionBehavior( NSInteger tag, NSWindowCollectionBehavior behavior )
WindowIsOpaque( NSInteger tag ) = Boolean
WindowSetOpaque( NSInteger tag, Boolean flag )
WindowHasShadow( NSInteger tag ) = Boolean
WindowSetHasShadow( NSInteger tag, Boolean flag )
WindowInvalidateShadow( NSInteger tag )
WindowAutorecalculatesContentBorderThickness( NSInteger tag, CGRectEdge edge ) = Boolean
WindowSetAutorecalculatesContentBorderThickness( NSInteger tag, Boolean flag, CGRectEdge edge )
WindowContentBorderThickness( NSInteger tag, CGRectEdge edge ) = CGFloat
WindowSetContentBorderThickness( NSInteger tag, CGFloat value, CGRectEdge edge )
WindowPreventsApplicationTerminationWhenModal( NSInteger tag ) = Boolean
WindowSetPreventsApplicationTerminationWhenModal( NSInteger tag, Boolean flag )
```

#### Info

```
WindowDefaultDepthLimit = NSWindowDepth
WindowNumber( NSInteger tag ) = NSInteger
WindowNumbersWithOptions( NSWindowNumberListOptions options ) = CFArrayRefarray of CFNumbers
WindowDeviceDescription( NSInteger tag ) = CFDictionaryRef
WindowCanBecomeVisibleWithoutLogin( NSInteger tag ) = Boolean
WindowSetCanBecomeVisibleWithoutLogin( NSInteger tag, Boolean flag )
WindowSharingType( NSInteger tag ) = NSWindowSharingType
WindowSetSharingType( NSInteger tag, NSWindowSharingType type )
WindowBackingType( NSInteger tag ) = NSBackingStoreType
WindowSetBackingType( NSInteger tag, NSBackingStoreType type )
WindowDepthLimit( NSInteger tag ) = NSWindowDepth
WindowSetDepthLimit( NSInteger tag, NSWindowDepth depth )
WindowHasDynamicDepthLimit( NSInteger tag ) = Boolean
```

#### Layout info

```
WindowContentRectForFrameRectAndStyle( CGRect r, NSInteger styleMask ) = CGRect
WindowFrameRectForContentRectAndStyle( CGRect r, NSInteger styleMask ) = CGRect
WindowMinFrameWidthWithTitle( CFStringRef title, NSInteger styleMask ) = CGFloat
WindowContentRectForFrameRect( NSInteger tag, CGRect r ) = CGRect
WindowFrameRectForContentRect( NSInteger tag, CGRect r ) = CGRect
```

#### Sheets

```
WindowAttachedSheet = NSInteger
WindowIsSheet( NSInteger tag ) = Boolean
WindowBeginSheet( NSInteger wndTag, NSInteger sheetTag )
WindowBeginCriticalSheet( NSInteger tag, NSInteger sheetTag )// macOS 10.9+
WindowEndSheet( NSInteger wndTag, NSInteger sheetTag )
WindowSheetParent( NSInteger tag ) = NSIntegermacOS 10.9+
WindowSheets( NSInteger tag ) = CFArrayRef// macOS 10.9+
```

#### Sizing

```
WindowFrame( NSInteger tag ) = CGRect
```

```
WindowSetFrame( NSInteger tag, CGRect frame )
WindowSetFrameOrigin( NSInteger tag, CGPoint pt )
WindowSetFrameTopLeftPoint( NSInteger tag, CGPoint pt )
WindowConstrainFrameRect( NSInteger tag, CGRect frame, ScreenRef screen )
WindowCascadeTopLeftFromPoint( NSInteger tag, CGPoint pt ) = CGPoint
WindowAspectRatio( NSInteger tag ) = CGSize
WindowSetAspectRatio( NSInteger tag )
WindowSetMinSize( NSInteger tag, CGSize size )
WindowSetMaxSize( NSInteger tag, CGSize size )
WindowIsZoomed( NSInteger tag ) = Boolean
WindowPerformZoom( NSInteger tag )
WindowZoom( NSInteger tag )
WindowResizeFlags( NSInteger tag ) = NSEventModifierFlags
WindowShowsResizeIndicator( NSInteger tag ) = Boolean
WindowSetShowsResizeIndicator( NSInteger tag, Boolean flag )
WindowResizeIncrements( NSInteger tag ) = CGSize
WindowSetResizeIncrements( NSInteger tag, CGSize size )
WindowInLiveResize( NSInteger tag ) = Boolean
```

#### Sizing content

```
WindowSetContentAspectRatio( NSInteger tag, CGSize size )
WindowContentMinSize( NSInteger tag ) = CGSize
WindowSetContentMinSize( NSInteger tag, CGSize size )
WindowContentSize( NSInteger tag ) = CGSize
WindowSetContentSize( NSInteger tag, CGSize size )
WindowContentMaxSize( NSInteger tag ) = CGSize
WindowSetContentMaxSize( NSInteger tag, CGSize size )
WindowContentResizeIncrements( NSInteger tag ) = CGSize
WindowSetContentResizeIncrements( NSInteger tag, CGSize size )
WindowMaxFullScreenContentSize( NSInteger tag ) = CGSizemacOS 10.11+
WindowSetMaxFullScreenContentSize( NSInteger tag, CGSize size )macOS 10.11+
WindowMinFullScreenContentSize( NSInteger tag ) = CGSizemacOS 10.11+
WindowSetMinFullScreenContentSize( NSInteger tag, CGSize size )macOS 10.11+
```

#### Layers

```
WindowOrderOut( NSInteger tag )
WindowOrderBack( NSInteger tag )
WindowOrderFront( NSInteger tag )
WindowOrderFrontRegardless( NSInteger tag )
WindowOrderRelativeTo( NSInteger wndTag, NSWindowOrderingMode order, NSInteger otherWndTag )
WindowLevel( NSInteger tag ) = NSWindowLevel
WindowSetLevel( NSInteger tag, NSWindowLevel level )
```

#### Visibility

```
WindowIsVisible( NSInteger tag ) = Boolean
WindowOcclusionState( NSInteger tag ) = NSWindowOcclusionStatemacOS 10.9+
```

#### Frame in user defaults

```
WindowRemoveFrameUsingName( CFStringRef name )
WindowSetFrameUsingName( NSInteger tag, CFStringRef name )
WindowSetFrameUsingNameForce( NSInteger tag, CFStringRef name, Boolean force )
WindowSaveFrameUsingName( NSInteger tag, CFStringRef name )
WindowFrameAutosaveName( NSInteger tag ) = CFStringRef
WindowSetFrameAutosaveName( NSInteger tag, CFStringRef name )
WindowStringWithSavedFrame( NSInteger tag ) = CFStringRef
WindowSetFrameFromString( NSInteger tag, CFStringRef string )
```

#### Key status

```
WindowIsKey( NSInteger tag ) = Boolean
WindowCanBecomeKey( NSInteger tag ) = Boolean
WindowMakeKey( NSInteger tag )
WindowMakeKeyAndOrderFront( NSInteger tag )
```

#### Main status

```
WindowIsMain( NSInteger tag ) = Boolean
WindowCanBecomeMain( NSInteger tag ) = Boolean
WindowMakeMain( NSInteger tag )
```

#### Toolbars

```
WindowToolbar( NSInteger tag ) = NSInteger
WindowSetToolbar( NSInteger wndTag, NSInteger toolbarTag )
WindowToggleToolbarShown( NSInteger tag )
WindowRunToolbarCustomizationPalette( NSInteger tag )
```

#### Attached windows

```
WindowChildWindows( NSInteger tag ) = CFArrayRef
WindowAddChildWindow( NSInteger parTag, NSInteger childTag, NSWindowOrderingMode ordered )
WindowRemoveChildWindow( NSInteger parTag, NSInteger childTag )
WindowParentWindow( NSInteger tag ) = NSInteger
```

#### Buffers

WindowFlush( NSInteger tag )

#### Window menu

WindowIsExcludedFromWindowsMenu( NSInteger tag ) = Boolean  
WindowSetExcludedFromWindowsMenu( NSInteger tag, Boolean flag )

#### Cursor rectangles

WindowAreCursorRectsEnabled( NSInteger tag ) = Boolean  
WindowEnableCursorRects( NSInteger tag )  
WindowDisableCursorRects( NSInteger tag )  
WindowDiscardCursorRects( NSInteger tag )  
WindowInvalidateCursorRectsForView( NSInteger tag, NSInteger vwTag )  
WindowResetCursorRects( NSInteger tag )

#### Titlebars

WindowStandardWindowButtonForStyleMask( NSWindowButton btn, NSUInteger styleMask ) = ptr  
WindowStandardWindowButton( NSInteger tag, NSWindowButton btn ) = ptr  
WindowTitlebarAppearsTransparent( NSInteger tag ) = Boolean macOS 10.10  
WindowSetTitlebarAppearsTransparent( NSInteger tag, Boolean flag )// macOS 10.10

#### Toolbar-titlebars

WindowAddTitlebarAccessoryView( NSInteger wndTag, NSInteger viewTag, NSLayoutAttribute layout )macOS 10.10+  
WindowInsertTitlebarAccessoryView( NSInteger wndTag, NSInteger viewTag, NSInteger index, NSLayoutAttribute layout )macOS 10.10+  
WindowRemoveTitlebarAccessoryView( NSInteger wndTag, NSInteger viewTag )macOS 10.10+

#### Tabs

WindowAllowsAutomaticWindowTabbing = Boolean// macOS 10.12+  
WindowUserTabbingPreference = NSWindowUserTabbingPreference// macOS 10.12+  
WindowTab( NSInteger tag ) = WindowTabRef// macOS 10.13+  
WindowTabbingIdentifier( NSInteger tag ) = CFStringRef// macOS 10.12+  
WindowAddTabbedWindow( NSInteger tag, NSInteger otherTag, NSWindowOrderingMode orderingMode )// macOS 10.12+  
WindowTabbingMode( NSInteger tag ) = NSWindowTabbingMode// macOS 10.12+  
WindowTabbedWindows( NSInteger tag ) = CFArrayRef// macOS 10.12+  
WindowSelectNextTab( NSInteger tag )// macOS 10.12+  
WindowSelectPreviousTab( NSInteger tag )// macOS 10.12+  
WindowMoveTabToNewWindow( NSInteger tag )// macOS 10.12+  
WindowToggleTabBar( NSInteger tag )// macOS 10.12+  
WindowToggleTabOverview( NSInteger tag )// macOS 10.13+  
WindowTabGroup( NSInteger tag ) = WindowTabGroupRef// macOS 10.13+

#### Tooltips

WindowSetAllowsToolTipsWhenApplicationIsInactive( NSInteger tag, Boolean flag)

#### Responders

WindowInitialFirstResponder( NSInteger tag ) = NSInteger  
WindowSetInitialFirstResponder( NSInteger wndTag, NSInteger responderTag )  
WindowFirstResponder( NSInteger tag ) = NSInteger  
WindowMakeFirstResponder( NSInteger wndTag, NSInteger viewTag )

#### Key view loop

WindowSelectKeyViewPrecedingView( NSInteger wndTag, NSInteger viewTag )  
WindowSelectKeyViewFollowingView( NSInteger wndTag, NSInteger viewTag )  
WindowSelectPreviousKeyView( NSInteger tag )  
WindowSelectNextKeyView( NSInteger tag )  
WindowKeyViewSelectionDirection( NSInteger tag ) = NSSelectionDirection  
WindowAutorecalculatesKeyViewLoop( NSInteger tag ) = Boolean  
WindowSetAutorecalculatesKeyViewLoop( NSInteger tag, Boolean flag )  
WindowRecalculateKeyViewLoop( NSInteger tag )

#### Mouse events

WindowAcceptsMouseMovedEvents( NSInteger tag ) = Boolean  
WindowSetAcceptsMouseMovedEvents( NSInteger tag, Boolean flag )  
WindowIgnoresMouseEvents( NSInteger tag ) = Boolean  
WindowSetIgnoresMouseEvents( NSInteger tag, Boolean flag )  
WindowMouseLocationOutsideOfEventStream( NSInteger tag ) = CGPoint  
WindowNumberAtPoint( CGPoint pt, NSInteger belowWindowNumber ) = NSInteger

#### Window restoration

WindowIsRestorable( NSInteger tag ) = Boolean  
WindowSetRestorable( NSInteger tag, Boolean flag )  
WindowRestorationClass( NSInteger tag ) = ClassRef  
WindowDisableSnapshotRestoration( NSInteger tag )  
WindowEnableShapshotRestoration( NSInteger tag )

#### Drawing

WindowDisplay( NSInteger tag )  
WindowDisplayIfNeeded( NSInteger tag )  
WindowViewsNeedDisplay( NSInteger tag ) = Boolean  
WindowSetViewsNeedDisplay( NSInteger tag, Boolean flag )  
WindowGraphicsContext( NSInteger tag ) = CGContextRef

```
WindowAllowsConcurrentViewDrawing( NSInteger tag ) = Boolean
WindowSetAllowsConcurrentViewDrawing( NSInteger tag, Boolean flag )
```

#### Animation

```
WindowAnimationBehavior( NSInteger tag ) = NSWindowAnimationBehavior
WindowSetAnimationBehavior( NSInteger tag, NSWindowAnimationBehavior behavior )
```

#### Dragging items

```
WindowRegisterForDraggedTypes( NSInteger tag, CFArrayRef types )
WindowUnregisterDraggedTypes( NSInteger tag )
```

#### Editing status

```
WindowDocumentEdited( NSInteger tag ) = Boolean
WindowSetDocumentEdited( NSInteger tag, Boolean flag )
```

#### Converting coordinates

```
WindowBackingScaleFactor( NSInteger tag ) = CGFloat
WindowBackingAlignedRect( NSInteger tag, CGRect r, NSAlignmentOptions options ) = CGRect
WindowConvertRectFromBacking( NSInteger tag, CGRect r ) = CGRect
WindowConvertRectToBacking( NSInteger tag, CGRect r ) = CGRect
WindowConvertRectToScreen( NSInteger tag, CGRect r ) = CGRect
WindowConvertRectFromScreen( NSInteger tag, CGRect r ) = CGRect
```

#### Titles

```
WindowTitle( NSInteger tag ) = CFStringRef
WindowSetTitle( NSInteger tag, CFStringRef title )
WindowTitleVisibility( NSInteger tag ) = NSWindowTitleVisibilitymacOS 10.10+
WindowSetTitleVisibility( NSInteger tag, NSWindowTitleVisibility visibility )
WindowSetTitleWithRepresentedFilename( NSInteger tag, CFStringRef filename )
WindowSetTitleWithRepresentedURL( NSInteger tag, CFURLRef url )
WindowRepresentedURL( NSInteger tag ) = CFURLRef
WindowSetRepresentedURL( NSInteger tag, CFURLRef url )
```

#### Screen info

```
WindowScreen( NSInteger tag ) = ScreenRef
WindowDeepestScreen( NSInteger tag ) = ScreenRef
WindowDisplaysWhenScreenProfileChanges( NSInteger tag ) = Boolean
WindowSetDisplaysWhenScreenProfileChanges( NSInteger tag, Boolean flag )
```

#### Moving

```
WindowIsMovableByWindowBackground( NSInteger tag ) = Boolean
WindowSetMovableByWindowBackground( NSInteger tag, Boolean flag )
WindowIsMovable( NSInteger tag ) = Boolean
WindowSetMovable( NSInteger tag, Boolean flag )
WindowCenter( NSInteger tag )
```

#### Closing

```
WindowPerformClose( NSInteger tag )
WindowClose( NSInteger tag )
    WindowIsReleasedWhenClosed( NSInteger tag ) = Boolean
WindowSetReleasedWhenClosed( NSInteger tag, Boolean flag )
```

#### Miniaturizing

```
WindowIsMiniaturized( NSInteger tag ) = Boolean
WindowPerformMiniaturize( NSInteger tag )
WindowMiniaturize( NSInteger tag )
WindowDeminiaturize( NSInteger tag )
WindowMiniwindowImage( NSInteger tag ) = ImageRef
WindowSetMiniwindowImage( NSInteger tag, ImageRef image )
WindowSetMiniwindowImageNamed( NSInteger tag, CFStringRef imageName )
WindowMiniwindowTitle( NSInteger tag ) = CFStringRef
WindowSetMiniwindowTitle( NSInteger tag, CFStringRef title )
```

#### Dock tile

```
WindowDockTile( NSInteger tag ) = DockTileRef
```

#### Printing

```
WindowPrint( NSInteger tag )
WindowDataWithEPSInsideRect( NSInteger tag, CGRect r ) = CFDataRef
WindowDataWithPDFInsideRect( NSInteger tag, CGRect r ) = CFDataRef
```

#### Scripting attributes

```
WindowHasCloseBox( NSInteger tag ) = Boolean
WindowHasTitleBar( NSInteger tag ) = Boolean
WindowSetIsMiniaturized( NSInteger tag, Boolean flag )
WindowSetIsVisible( NSInteger tag, Boolean flag )
WindowSetIsZoomed( NSInteger tag, Boolean flag )
```

#### Ordered indices

```
WindowOrderedIndex( NSInteger tag ) = NSInteger
WindowSetOrderedIndex( NSInteger tag, NSInteger index )
```

#### Instance properties

```
WindowIsFloatingPanel( NSInteger tag ) = Boolean
WindowIsMiniaturizable( NSInteger tag ) = Boolean
WindowIsModalPanel( NSInteger tag ) = Boolean
WindowIsResizable( NSInteger tag ) = Boolean
WindowStyleMask( NSInteger tag ) = NSUInteger
WindowIsZoomable( NSInteger tag ) = Boolean
```

#### Instance methods

```
WindowMergeAllWindows( NSInteger tag )// macOS 10.12+
WindowSetFrameAutosaveName( NSInteger tag, CFStringRef name )
```

#### Appearance customization

```
WindowAppearance( NSInteger tag ) = AppearanceRef// macOS 10.9+
WindowEffectiveAppearance( NSInteger tag ) = AppearanceRef// macOS 10.9+
```

#### Convenience

```
WindowContentRect( NSInteger tag ) = CGRect
toolbox WindowAddSubview( NSInteger tag, ViewRef vwRef )
WindowViewWithTag( NSInteger wndTag, NSInteger viewTag ) = ptr
WindowRecalculateKeyViewLoopByTagOrder( NSInteger wndTag )
WindowSubclassContentView( NSInteger wndTag )
WindowSetDelegateCallback( NSInteger tag, ptr callback, ptr userData )
WindowRegisterForDraggedFileTypes( NSInteger tag, CFTypeRef fileTypes )
WindowRegisterForDraggedFileExtensions( NSInteger tag, CFTypeRef extensions )
WindowDragInside( NSInteger tag ) = Boolean
WindowSetDragInside( NSInteger tag, Boolean flag )
```

#### Custom

```
WindowSetOutput( NSInteger tag )
WindowProperty( NSInteger tag, CFStringRef key ) = CFTypeRef
WindowSetProperty( NSInteger tag, CFStringRef key, CFTypeRef value )
WindowRemoveProperty( NSInteger tag, CFStringRef key )
WindowRemoveAllProperties( NSInteger tag )
WindowCascadeTopLeftFromWindow( NSInteger tag, NSInteger otherTag )
WindowCascadeTopLeftFromNextWindow( NSInteger tag )
WindowEndSheetAlert( NSInteger tag, NSInteger alertTag )// macOS 10.9+
```

## Apple documentation

[NSWindow](#)



---

## WindowTab

### Functions

Instance properties

```
WindowTabAccessoryView( WindowTabRef ref ) = NSInteger
WindowTabSetAccessoryView( WindowTabRef ref, NSInteger tag )
WindowTabAttributedTitle( WindowTabRef ref ) = CFAttributedStringRef
WindowTabSetAttributedTitle( WindowTabRef ref, CFAttributedStringRef aString )
WindowTabTitle( WindowTabRef ref ) = CFStringRef
WindowTabSetTitle( WindowTabRef ref, CFStringRef title )
WindowTabToolTip( WindowTabRef ref ) = CFStringRef
WindowTabSetToolTip( WindowTabRef ref, CFStringRef toolTip )
```

### Apple documentation

[NSWindowTab](#)



---

### WindowTabGroup

#### Functions

Instance properties

```
WindowTabGroupIdentifier( WindowTabGroupRef ref ) = CFStringRef
WindowTabGroupIsOverviewVisible( WindowTabGroupRef ref ) = Boolean
WindowTabGroupSetOverviewVisible( WindowTabGroupRef ref, Boolean flag )
WindowTabGroupIsTabBarVisible( WindowTabGroupRef ref ) = Boolean
WindowTabGroupSelectedWindow( WindowTabGroupRef ref ) = NSInteger
WindowTabGroupSetSelectedWindow( WindowTabGroupRef ref, NSInteger tag )
WindowTabGroupWindows( WindowTabGroupRef ref ) = CFArrayRef
```

Instance methods

```
WindowTabGroupAddWindow( WindowTabGroupRef ref, NSInteger tag )
WindowTabGroupInsertWindow( WindowTabGroupRef ref, NSInteger tag, NSInteger index )
WindowTabGroupRemoveWindow( WindowTabGroupRef ref, NSInteger tag )
```

#### Apple documentation

[NSWindowTabGroup](#)



---

## WindowMenu

statement

### Syntax

**windowmenu** *menuIndex*

### Description

Builds the default window menu. Requires the Cocoa runtime ([CocoaInit](#)).





## Workspace

### Functions

Shared

```
WorkspaceShared = WorkspaceRef
```

Notification center

```
WorkspaceNotificationCenter = NotificationCenterRef
WorkspaceNotificationCenterAddObserver( ptr callback, CFStringRef name, CFTypeRef obj )
WorkspaceNotificationCenterRemoveObserver( ptr callback, CFStringRef name )
WorkspaceNotificationCenterPostNotification( NotificationRef note )
WorkspaceNotificationCenterPostNotificationName( CFStringRef name, CFTypeRef obj, CFDictionaryRef userInfo )
```

Opening files

```
WorkspaceOpenURLWithOptions( CFURLRef url, NSWspaceLaunchOptions options, CFDictionaryRef configuration, ErrorRef
*err ) = RunningApplicationRef// macOS 10.10+
WorkspaceOpenURL( CFURLRef url ) = Boolean
WorkspaceOpenURLWithApplication( CFURLRef url, CFStringRef appName, Boolean deactivate ) = Boolean
WorkspaceOpenURLsWithApplicationAtURL( CFArrayRef urls, CFURLRef appURL, NSWspaceLaunchOptions options,
CFDictionaryRef configuration ) = Boolean
WorkspaceOpenURLWithApplicationAtURL( CFURLRef url, CFURLRef appURL, NSWspaceLaunchOptions options,
CFDictionaryRef configuration ) = Boolean
```

Manipulating applications

```
WorkspaceLaunchApplication( CFStringRef appName ) = Boolean
WorkspaceLaunchApplicationAtURL( CFURLRef url, NSWspaceLaunchOptions options, CFDictionaryRef configuration,
ErrorRef *err ) = Boolean
WorkspaceHideOtherApplications
```

Manipulating files

```
WorkspaceDuplicateURLs( CFArrayRef urls )
WorkspaceRecycleURL( CFURLRef url)
WorkspaceRecycleURLs( CFArrayRef urls )
WorkspaceActivateFileViewerSelectingURLs( CFArrayRef fileURLs )
```

UTI info

```
WorkspaceTypeOfFileAtURL( CFURLRef url ) = CFStringRef
WorkspaceLocalizedDescriptionForType( CFStringRef type ) = CFStringRef
WorkspacePreferredFilenameExtensionForType( CFStringRef typeName ) = CFStringRef
WorkspaceFilenameExtensionIsValidForType( CFStringRef filenameExtension, CFStringRef typeName ) = Boolean
WorkspaceFileTypeConformToType( CFStringRef firstTypeName, CFStringRef secondTypeName ) = Boolean
WorkspaceFileTypeConformsToType( CFStringRef firstTypeName, CFStringRef secondTypeName ) = Boolean
WorkspaceURLForApplicationWithBundleIdentifier( CFStringRef identifier ) = CFURLRef
WorkspaceURLForApplication( CFStringRef name ) = CFURLRef
```

Requesting info

```
WorkspaceGetInfoForFileAtURL( CFURLRef url, CFStringRef *appName, CFStringRef *type ) = Boolean
WorkspaceURLForApplicationToOpenURL( CFURLRef url ) = CFURLRef
WorkspaceGetFileSystemInfoForURL( CFURLRef url, Boolean *isRemovable, Boolean *isWritable, Boolean *isUnmountable,
CFStringRef *description, CFStringRef *type ) = Boolean
WorkspaceIsFilePackageAtURL( CFURLRef url ) = Boolean
WorkspaceFrontmostApplication = RunningApplicationRef
WorkspaceRunningApplications = CFArrayRef
WorkspaceMenuBarOwningApplication = RunningApplicationRef
```

Icons

```
WorkspaceIconForFileAtURL( CFURLRef url ) = ImageRef
WorkspaceIconForFileType( CFStringRef type ) = ImageRef
```

Unmount

```
WorkspaceUnmountAndEjectDeviceAtURL( CFURLRef url, ErrorRef *err ) = Boolean
```

Desktop image

```
WorkspaceDesktopImageURLForScreen( ScreenRef screen ) = CFURLRef
WorkspaceSetDesktopImageURLForScreen( CFURLRef url, ScreenRef screen, CFDictionaryRef options, ErrorRef *err )
WorkspaceDesktopImageOptionsForScreen( ScreenRef screen ) = CFDictionaryRef
```

Finder spotlight searches

```
WorkspaceShowSearchResultsForQueryString( CFStringRef string ) = Boolean
```

Finder file labels

```
WorkspaceFileLabelColors = CFArrayRef
```

`WorkspaceFileLabels = CFArrayRef`

Tracking changes to file system

`WorkspaceNoteFileSystemChangedAtURL( CFURLRef url )`

Instance methods

`WorkspaceRequestAuthorizationOfType( NSWorkspaceAuthorizationType type, ptr completionHandler, ptr userData )//`  
macOS 10.14+

## **Apple documentation**

[NSWorkspace](#)