NSResponder Class Reference

Developer

Contents

NSResponder Class Reference 7
Overview 7
Tasks 8
Changing the First Responder 8
Managing the Next Responder 8
Responding to Mouse Events 9
Responding to Key Events 9
Responding to Other Kinds of Events 10
Responding to Action Messages 10
Handling Window Restoration 18
Presenting and Customizing Error Information 18
Dispatching Messages 18
Managing a Responder's Menu 18
Updating the Services Menu 19
Getting the Undo Manager 19
Testing Events 19
Terminating the Responder Chain 19
Setting the Interface Style 19
Touch and Gesture Events 19
Setting the Writing Direction 20
Performing Text Find Actions 21
21
Class Methods 21
restorableStateKeyPaths 21
Instance Methods 22
acceptsFirstResponder 22
becomeFirstResponder 22
beginGestureWithEvent: 23
cancelOperation: 23
capitalizeWord: 24
centerSelectionInVisibleArea: 25
changeCaseOfLetter: 25
complete: 26
cursorUpdate: 26

© 2011 Apple Inc. All Rights Reserved.

deleteBackward: 27
deleteBackwardByDecomposingPreviousCharacter: 27
deleteForward: 28
deleteToBeginningOfLine: 28
deleteToBeginningOfParagraph: 29
deleteToEndOfLine: 29
deleteToEndOfParagraph: 30
deleteToMark: 30
deleteWordBackward: 31
deleteWordForward: 32
doCommandBySelector: 32
encodeRestorableStateWithCoder: 33
endGestureWithEvent: 33
flagsChanged: 34
flushBufferedKeyEvents 34
helpRequested: 35
indent: 35
insertBacktab: 36
insertContainerBreak: 36
insertDoubleQuoteIgnoringSubstitution: 37
insertLineBreak: 37
insertNewline: 38
insertNewlineIgnoringFieldEditor: 38
insertParagraphSeparator: 39
insertSingleQuoteIgnoringSubstitution: 39
insertTab: 40
insertTabIgnoringFieldEditor: 40
insertText: 41
interpretKeyEvents: 41
invalidateRestorableState 42
keyDown: 42
keyUp: 43
lowercaseWord: 43
magnifyWithEvent: 44
makeBaseWritingDirectionLeftToRight: 44
makeBaseWritingDirectionNatural: 45
makeBaseWritingDirectionRightToLeft: 46
makeTextWritingDirectionLeftToRight: 46
makeTextWritingDirectionNatural: 47

2011-06-06 | © 2011 Apple Inc. All Rights Reserved.

makeTextWritingDirectionRightToLeft: 47
menu 48
mouseDown: 49
mouseDragged: 49
mouseEntered: 50
mouseExited: 50
mouseMoved: 50
mouseUp: 51
moveBackward: 51
moveBackwardAndModifySelection: 52
moveDown: 52
moveDownAndModifySelection: 53
moveForward: 54
moveForwardAndModifySelection: 54
moveLeft: 55
moveLeftAndModifySelection: 55
moveParagraphBackwardAndModifySelection: 56
moveParagraphForwardAndModifySelection: 56
moveRight: 57
moveRightAndModifySelection: 57
moveToBeginningOfDocument: 58
moveToBeginningOfDocumentAndModifySelection: 59
moveToBeginningOfLine: 59
moveToBeginningOfLineAndModifySelection: 59
moveToBeginningOfParagraph: 60
moveToBeginningOfParagraphAndModifySelection: 60
moveToEndOfDocument: 61
moveToEndOfDocumentAndModifySelection: 61
moveToEndOfLine: 62
moveToEndOfLineAndModifySelection: 62
moveToEndOfParagraph: 63
moveToEndOfParagraphAndModifySelection: 63
moveToLeftEndOfLine: 63
moveToLeftEndOfLineAndModifySelection: 64
moveToRightEndOfLine: 64
moveToRightEndOfLineAndModifySelection: 65
moveUp: 65
moveUpAndModifySelection: 66
moveWordBackward: 66

© 2011 Apple Inc. All Rights Reserved.

moveWordBackwardAndModifySelection: 67
moveWordForward: 67
moveWordForwardAndModifySelection: 68
moveWordLeft: 68
moveWordLeftAndModifySelection: 69
moveWordRight: 70
moveWordRightAndModifySelection: 70
nextResponder 71
noResponderFor: 71
otherMouseDown: 72
otherMouseDragged: 72
otherMouseUp: 73
pageDown: 73
pageDownAndModifySelection: 74
pageUp: 74
pageUpAndModifySelection: 75
performKeyEquivalent: 75
performTextFinderAction: 76
presentError: 77
presentError:modalForWindow:delegate:didPresentSelector:contextInfo: 78
resignFirstResponder 79
restoreStateWithCoder: 80
rightMouseDown: 80
rightMouseDragged: 81
rightMouseUp: 81
rotateWithEvent: 82
scrollLineDown: 82
scrollLineUp: 83
scrollPageDown: 83
scrollPageUp: 84
scrollToBeginningOfDocument: 85
scrollToEndOfDocument: 85
scrollWheel: 85
selectAll: 86
selectLine: 86
selectParagraph: 87
selectToMark: 87
selectWord: 88
setMark: 88

2011-06-06 | © 2011 Apple Inc. All Rights Reserved.

setMenu: 89
setNextResponder: 89
shouldBeTreatedAsInkEvent: 90
showContextHelp: 91
supplementalTargetForAction:sender: 91
swapWithMark: 92
swipeWithEvent: 92
tabletPoint: 93
tabletProximity: 93
touchesBeganWithEvent: 94
touchesCancelledWithEvent: 95
touchesEndedWithEvent: 95
touchesMovedWithEvent: 96
transpose: 96
transposeWords: 97
tryToPerform:with: 97
undoManager 98
uppercaseWord: 98
validateProposedFirstResponder:forEvent: 99
validRequestorForSendType:returnType: 100
wantsForwardedScrollEventsForAxis: 101
willPresentError: 101
yank: 102

Deprecated NSResponder Methods 104
Deprecated in OS X v10.8 104
interfaceStyle 104
performMnemonic: 104
setInterfaceStyle: 105

Document Revision History 107

© 2011 Apple Inc. All Rights Reserved.

NSResponder Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in OS X v10.0 and later.
Companion guide	Cocoa Event Handling Guide
Declared in	NSInterfaceStyle.h NSResponder.h NSWindowRestoration.h
Related sample code	AnimatedTableView GLFullScreen GLUT LayerBackedOpenGLView SourceView

Overview

`NSResponder` is an abstract class that forms the basis of event and command processing in the Application Kit. The core classes—`NSApplication`, `NSWindow`, and `NSView`—inherit from `NSResponder`, as must any class that handles events. The responder model is built around three components: event messages, action messages, and the responder chain.

Starting with OS X v10.4, `NSResponder` plays an important role in the presentation of error information. The default implementations of the `presentError:` (page 77) and `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 78) methods send `willPresentError:` (page 101) to `self`, thereby giving subclasses the opportunity to customize the localized

© 2011 Apple Inc. All Rights Reserved.

information presented in error alerts. NSResponder then forwards the message to the next responder, passing it the customized NSError object. The exact path up the modified responder chain depends on the type of application window:

- Windows owned by document: view to superviews to window to window controller to document object to document controller to the application object
- Windows with window controllers but no documents: view to superviews to window to window controller to the application object
- Windows with no window controllers: view to superviews to window to the application object

NSApplication displays a document-modal error alert and, if the error object has a recovery attempter, gives it a chance to recover from the error. (A recovery attempter is an object that conforms to the NSErrorRecoveryAttempting informal protocol.)

Tasks

Changing the First Responder

– acceptsFirstResponder (page 22)
  Overridden by subclasses to return YES if the receiver accepts first responder status.

– becomeFirstResponder (page 22)
  Notifies the receiver that it's about to become first responder in its NSWindow.

– resignFirstResponder (page 79)
  Notifies the receiver that it's been asked to relinquish its status as first responder in its window.

– validateProposedFirstResponder:forEvent: (page 99)
  Allow controls to determine when they should become first responder.

Managing the Next Responder

– setNextResponder: (page 89)
  Sets the receiver's next responder.

– nextResponder (page 71)
  Returns the receiver's next responder, or nil if it has none.

© 2011 Apple Inc. All Rights Reserved.

Responding to Mouse Events

- `mouseDown:` (page 49)
  Informs the receiver that the user has pressed the left mouse button.
- `mouseDragged:` (page 49)
  Informs the receiver that the user has moved the mouse with the left button pressed.
- `mouseUp:` (page 51)
  Informs the receiver that the user has released the left mouse button.
- `mouseMoved:` (page 50)
  Informs the receiver that the mouse has moved.
- `mouseEntered:` (page 50)
  Informs the receiver that the cursor has entered a tracking rectangle.
- `mouseExited:` (page 50)
  Informs the receiver that the cursor has exited a tracking rectangle.
- `rightMouseDown:` (page 80)
  Informs the receiver that the user has pressed the right mouse button.
- `rightMouseDragged:` (page 81)
  Informs the receiver that the user has moved the mouse with the right button pressed .
- `rightMouseUp:` (page 81)
  Informs the receiver that the user has released the right mouse button.
- `otherMouseDown:` (page 72)
  Informs the receiver that the user has pressed a mouse button other than the left or right one.
- `otherMouseDragged:` (page 72)
  Informs the receiver that the user has moved the mouse with a button other than the left or right button pressed.
- `otherMouseUp:` (page 73)
  Informs the receiver that the user has released a mouse button other than the left or right button.

Responding to Key Events

- `keyDown:` (page 42)
  Informs the receiver that the user has pressed a key.
- `keyUp:` (page 43)
  Informs the receiver that the user has released a key.

© 2011 Apple Inc. All Rights Reserved.

– `interpretKeyEvents:` (page 41)

Invoked by subclasses from their `keyDown:` (page 42) method to handle a series of key events.

– `performKeyEquivalent:` (page 75)

Overridden by subclasses to handle a key equivalent.

– `flushBufferedKeyEvents` (page 34)

Overridden by subclasses to clear any unprocessed key events.

– `performMnemonic:` (page 104) Deprecated in OS X v10.8

Overridden by subclasses to handle a mnemonic.

Responding to Other Kinds of Events

– `cursorUpdate:` (page 26)

Informs the receiver that the mouse cursor has moved into a cursor rectangle.

– `flagsChanged:` (page 34)

Informs the receiver that the user has pressed or released a modifier key (Shift, Control, and so on).

– `tabletPoint:` (page 93)

Informs the receiver that a tablet-point event has occurred.

– `tabletProximity:` (page 93)

Informs the receiver that a tablet-proximity event has occurred.

– `helpRequested:` (page 35)

Displays context-sensitive help for the receiver if such exists; otherwise passes this message to the next responder.

– `scrollWheel:` (page 85)

Informs the receiver that the mouse's scroll wheel has moved.

Responding to Action Messages

– `cancelOperation:` (page 23)

Implemented by subclasses to cancel the current operation.

– `capitalizeWord:` (page 24)

Implemented by subclasses to capitalize the word or words surrounding the insertion point or selection, expanding the selection if necessary.

– `centerSelectionInVisibleArea:` (page 25)

Implemented by subclasses to scroll the selection, whatever it is, inside its visible area.

© 2011 Apple Inc. All Rights Reserved.

- `changeCaseOfLetter:` (page 25)
  Implemented by subclasses to change the case of a letter or letters in the selection, perhaps by opening a panel with capitalization options or by cycling through possible case combinations.
- `complete:` (page 26)
  Implemented by subclasses to complete an operation in progress or a partially constructed element.
- `deleteBackward:` (page 27)
  Implemented by subclasses to delete the selection, if there is one, or a single element backward from the insertion point (a letter or character in text, for example).
- `deleteBackwardByDecomposingPreviousCharacter:` (page 27)
  Implemented by subclasses to delete the selection, if there is one, or a single character backward from the insertion point.
- `deleteForward:` (page 28)
  Implemented by subclasses to delete the selection, if there is one, or a single element forward from the insertion point (a letter or character in text, for example).
- `deleteToBeginningOfLine:` (page 28)
  Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the beginning of a line (typically of text).
- `deleteToBeginningOfParagraph:` (page 29)
  Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the beginning of a paragraph of text.
- `deleteToEndOfLine:` (page 29)
  Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the end of a line (typically of text).
- `deleteToEndOfParagraph:` (page 30)
  Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the end of a paragraph of text.
- `deleteToMark:` (page 30)
  Implemented by subclasses to delete the selection, if there is one, or all items from the insertion point to a previously placed mark, including the selection itself if not empty.
- `deleteWordBackward:` (page 31)
  Implemented by subclasses to delete the selection, if there is one, or a single word backward from the insertion point.
- `deleteWordForward:` (page 32)
  Implemented by subclasses to delete the selection, if there is one, or a single word forward from the insertion point.

© 2011 Apple Inc. All Rights Reserved.

- `indent:` (page 35)
  Implemented by subclasses to indent the selection or the insertion point if there is no selection.
- `insertBacktab:` (page 36)
  Implemented by subclasses to handle a backward tab.
- `insertContainerBreak:` (page 36)
  Implemented by subclasses to insert a container break (typically a page break) at the insertion point or selection, deleting the selection if there is one.
- `insertLineBreak:` (page 37)
  Implemented by subclasses to insert a line break (as distinguished from a paragraph break) at the insertion point or selection, deleting the selection if there is one.
- `insertNewline:` (page 38)
  Implemented by subclasses to insert a newline character at the insertion point or selection, deleting the selection if there is one, or to end editing if the receiver is a text field or other field editor.
- `insertNewlineIgnoringFieldEditor:` (page 38)
  Implemented by subclasses to insert a line-break character at the insertion point or selection, deleting the selection if there is one.
- `insertParagraphSeparator:` (page 39)
  Implemented by subclasses to insert a paragraph separator at the insertion point or selection, deleting the selection if there is one.
- `insertTab:` (page 40)
  Implemented by subclasses to insert a tab character at the insertion point or selection, deleting the selection if there is one, or to end editing if the receiver is a text field or other field editor.
- `insertTabIgnoringFieldEditor:` (page 40)
  Implemented by subclasses to insert a tab character at the insertion point or selection, deleting the selection if there is one.
- `insertDoubleQuoteIgnoringSubstitution:` (page 37)
  Implemented by subclasses to insert a double quote character at the insertion point without interference by automatic quote correction.
- `insertSingleQuoteIgnoringSubstitution:` (page 39)
  Implemented by subclasses to insert a single quote character at the insertion point without interference by automatic quote correction.
- `insertText:` (page 41)
  Overridden by subclasses to insert the supplied string at the insertion point or selection, deleting the selection if there is one.

© 2011 Apple Inc. All Rights Reserved.

- `lowercaseWord:` (page 43)
  Implemented by subclasses to make lowercase every letter in the word or words surrounding the insertion point or selection, expanding the selection if necessary.
- `moveBackward:` (page 51)
  Implemented by subclasses to move the selection or insertion point one element or character backward.
- `moveBackwardAndModifySelection:` (page 52)
  Implemented by subclasses to expand or reduce either end of the selection backward by one element or character.
- `moveParagraphForwardAndModifySelection:` (page 56)
  Implemented by subclasses to move the selection or insertion point to the beginning of the next paragraph, expanding or reducing the current selection.
- `moveParagraphBackwardAndModifySelection:` (page 56)
  Implemented by subclasses to move the selection or insertion point to the beginning of the previous paragraph, expanding or reducing the current selection.
- `moveToBeginningOfDocumentAndModifySelection:` (page 59)
  Implemented by subclasses to move the selection or insertion point to the beginning of the document, expanding or reducing the current selection.
- `moveToEndOfDocumentAndModifySelection:` (page 61)
  Implemented by subclasses to move the selection or insertion point to the end of the document, expanding or reducing the current selection.
- `moveToBeginningOfLineAndModifySelection:` (page 59)
  Implemented by subclasses to move the selection or insertion point to the beginning of the line, expanding or reducing the current selection.
- `moveToEndOfLineAndModifySelection:` (page 62)
  Implemented by subclasses to move the selection or insertion point to the end of the line, expanding or reducing the current selection.
- `moveToBeginningOfParagraphAndModifySelection:` (page 60)
  Implemented by subclasses to move the selection or insertion point to the beginning of the current paragraph, expanding or reducing the current selection.
- `moveToEndOfParagraphAndModifySelection:` (page 63)
  Implemented by subclasses to move the selection or insertion point to the end of the current paragraph, expanding or reducing the current selection.
- `moveToLeftEndOfLine:` (page 63)
  Implemented by subclasses to move the selection or insertion point to the left end of the line.

© 2011 Apple Inc. All Rights Reserved.

- `moveToLeftEndOfLineAndModifySelection:` (page 64)
  Implemented by subclasses to move the selection or insertion point to the left end of the line, expanding or contracting the selection as required.
- `moveToRightEndOfLine:` (page 64)
  Implemented by subclasses to move the selection or insertion point to the right end of the line
- `moveToRightEndOfLineAndModifySelection:` (page 65)
  Implemented by subclasses to move the selection or insertion point to the right end of the line, expanding or contracting the selection as required.
- `moveDown:` (page 52)
  Implemented by subclasses to move the selection or insertion point one element or character down.
- `moveDownAndModifySelection:` (page 53)
  Implemented by subclasses to expand or reduce the top or bottom end of the selection downward by one element, character, or line (whichever is appropriate for text direction).
- `moveForward:` (page 54)
  Implemented by subclasses to move the selection or insertion point one element or character forward.
- `moveForwardAndModifySelection:` (page 54)
  Implemented by subclasses to expand or reduce either end of the selection forward by one element or character.
- `moveLeft:` (page 55)
  Implemented by subclasses to move the selection or insertion point one element or character to the left.
- `moveLeftAndModifySelection:` (page 55)
  Implemented by subclasses to expand or reduce either end of the selection to the left (display order) by one element or character.
- `moveRight:` (page 57)
  Implemented by subclasses to move the selection or insertion point one element or character to the right.
- `moveRightAndModifySelection:` (page 57)
  Implemented by subclasses to expand or reduce either end of the selection to the right (display order) by one element or character.
- `moveToBeginningOfDocument:` (page 58)
  Implemented by subclasses to move the selection to the first element of the document or the insertion point to the beginning.
- `moveToBeginningOfLine:` (page 59)
  Implemented by subclasses to move the selection to the first element of the selected line or the insertion point to the beginning of the line.

© 2011 Apple Inc. All Rights Reserved.

- `moveToBeginningOfParagraph:` (page 60)
  Implemented by subclasses to move the insertion point to the beginning of the selected paragraph.
- `moveToEndOfDocument:` (page 61)
  Implemented by subclasses to move the selection to the last element of the document or the insertion point to the end.
- `moveToEndOfLine:` (page 62)
  Implemented by subclasses to move the selection to the last element of the selected line or the insertion point to the end of the line.
- `moveToEndOfParagraph:` (page 63)
  Implemented by subclasses to move the insertion point to the end of the selected paragraph.
- `moveUp:` (page 65)
  Implemented by subclasses to move the selection or insertion point one element or character up.
- `moveUpAndModifySelection:` (page 66)
  Implemented by subclasses to expand or reduce the top or bottom end of the selection upward by one element, character, or line (whichever is appropriate for text direction).
- `moveWordBackward:` (page 66)
  Implemented by subclasses to move the selection or insertion point one word backward.
- `moveWordBackwardAndModifySelection:` (page 67)
  Implemented by subclasses to expand or reduce either end of the selection backward by one whole word.
- `moveWordForward:` (page 67)
  Implemented by subclasses to move the selection or insertion point one word forward, in logical order.
- `moveWordForwardAndModifySelection:` (page 68)
  Implemented by subclasses to expand or reduce either end of the selection forward by one whole word.
- `moveWordLeft:` (page 68)
  Implemented by subclasses to move the selection or insertion point one word to the left, in display order.
- `moveWordRight:` (page 70)
  Implemented by subclasses to move the selection or insertion point one word right.
- `moveWordRightAndModifySelection:` (page 70)
  Implemented by subclasses to expand or reduce either end of the selection to the right by one whole word.
- `moveWordLeftAndModifySelection:` (page 69)
  Implemented by subclasses to expand or reduce either end of the selection left by one whole word in display order.

© 2011 Apple Inc. All Rights Reserved.

- `pageDown:` (page 73)
  Implemented by subclasses to scroll the receiver down (or back) one page in its scroll view, also moving the insertion point to the top of the newly displayed page.
- `pageDownAndModifySelection:` (page 74)
  Implemented by subclasses to scroll the receiver down (or back) one page in its scroll view, also moving the insertion point to the top of the newly displayed page. The selection is expanded or contracted as required.
- `pageUp:` (page 74)
  Implemented by subclasses to scroll the receiver up (or forward) one page in its scroll view, also moving the insertion point to the top of the newly displayed page.
- `pageUpAndModifySelection:` (page 75)
  Implemented by subclasses to scroll the receiver up (or forward) one page in its scroll view, also moving the insertion point to the top of the newly displayed page. The selection is expanded or contracted as necessary.
- `scrollToBeginningOfDocument:` (page 85)
  Implemented by subclasses to scroll the receiver to the beginning of the document, without changing the selection.
- `scrollToEndOfDocument:` (page 85)
  Implemented by subclasses to scroll the receiver to the end of the document, without changing the selection.
- `scrollLineUp:` (page 83)
  Implemented by subclasses to scroll the receiver one line up in its scroll view, without changing the selection.
- `scrollLineDown:` (page 82)
  Implemented by subclasses to scroll the receiver one line down in its scroll view, without changing the selection.
- `scrollPageUp:` (page 84)
  Implemented by subclasses to scroll the receiver one page up in its scroll view, without changing the selection.
- `scrollPageDown:` (page 83)
  Implemented by subclasses to scroll the receiver one page down in its scroll view, without changing the selection.
- `selectAll:` (page 86)
  Implemented by subclasses to select all selectable elements.

2011-06-06 | © 2011 Apple Inc. All Rights Reserved.

- `selectLine:` (page 86)
  Implemented by subclasses to select all elements in the line or lines containing the selection or insertion point.
- `selectParagraph:` (page 87)
  Implemented by subclasses to select all paragraphs containing the selection or insertion point.
- `selectToMark:` (page 87)
  Implemented by subclasses to select all items from the insertion point or selection to a previously placed mark, including the selection itself if not empty.
- `selectWord:` (page 88)
  Implemented by subclasses to extend the selection to the nearest word boundaries outside it (up to, but not including, word delimiters).
- `setMark:` (page 88)
  Implemented by subclasses to set a mark at the insertion point or selection, which is used by `deleteToMark:` (page 30) and `selectToMark:` (page 87).
- `showContextHelp:` (page 91)
  Implemented by subclasses to invoke the help system, displaying information relevant to the receiver and its current state.
- `swapWithMark:` (page 92)
  Swaps the mark and the selection or insertion point, so that what was marked is now the selection or insertion point, and what was the insertion point or selection is now the mark.
- `transpose:` (page 96)
  Transposes the characters to either side of the insertion point and advances the insertion point past both of them. Does nothing to a selected range of text.
- `transposeWords:` (page 97)
  Transposes the words to either side of the insertion point and advances the insertion point past both of them. Does nothing to a selected range of text.
- `uppercaseWord:` (page 98)
  Implemented by subclasses to make uppercase every letter in the word or words surrounding the insertion point or selection, expanding the selection if necessary.
- `yank:` (page 102)
  Replaces the insertion point or selection with text from the kill buffer.
- `supplementalTargetForAction:sender:` (page 91)
  This method is used in the process of finding a target for an action method.

© 2011 Apple Inc. All Rights Reserved.

Handling Window Restoration

– `encodeRestorableStateWithCoder:` (page 33)
Saves the interface-related state of the responder.

– `restoreStateWithCoder:` (page 80)
Restores the interface-related state of the responder.

+ `restorableStateKeyPaths` (page 21)
Returns an array of key paths representing the restorable attributes of the responder.

– `invalidateRestorableState` (page 42)
Marks the responder's interface-related state as dirty.

Presenting and Customizing Error Information

– `presentError:` (page 77)
Presents an error alert to the user as an application-modal dialog.

– `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 78)
Presents an error alert to the user as a document-modal sheet attached to document window.

– `willPresentError:` (page 101)
Implemented by subclasses to return a custom version of the supplied error object that is more suitable for presentation in alert sheets and dialogs.

Dispatching Messages

– `doCommandBySelector:` (page 32)
Attempts to perform the indicated command.

– `tryToPerform:with:` (page 97)
Attempts to perform the action indicated method with a specified argument.

Managing a Responder's Menu

– `setMenu:` (page 89)
Sets the receiver's menu.

– `menu` (page 48)
Returns the receiver's menu.

© 2011 Apple Inc. All Rights Reserved.

Updating the Services Menu

– `validRequestorForSendType:returnType:` (page 100)
Overridden by subclasses to determine what services are available.

Getting the Undo Manager

– `undoManager` (page 98)
Returns the undo manager for this responder.

Testing Events

– `shouldBeTreatedAsInkEvent:` (page 90)
Returns `YES` if the specified event should be treated as an ink event, `NO` if it should be treated as a mouse event.

Terminating the Responder Chain

– `noResponderFor:` (page 71)
Handles the case where an event or action message falls off the end of the responder chain.

Setting the Interface Style

– `interfaceStyle` (page 104) Deprecated in OS X v10.8
Returns the receiver's interface style.

– `setInterfaceStyle:` (page 105) Deprecated in OS X v10.8
Sets the receiver's style to the style specified by `interfaceStyle`, such as `NSMacintoshInterfaceStyle` or `NSWindows95InterfaceStyle`.

Touch and Gesture Events

– `beginGestureWithEvent:` (page 23)
Informs the receiver that the user has begun a touch gesture.

– `endGestureWithEvent:` (page 33)
Informs the receiver that the user has ended a touch gesture.

© 2011 Apple Inc. All Rights Reserved.

- `magnifyWithEvent:` (page 44)
  Informs the receiver that the user has begun a pinch gesture.
- `rotateWithEvent:` (page 82)
  Informs the receiver that the user has begun a rotation gesture.
- `swipeWithEvent:` (page 92)
  Informs the receiver that the user has begun a swipe gesture.
- `touchesBeganWithEvent:` (page 94)
  Informs the receiver that new set of touches has been recognized.
- `touchesMovedWithEvent:` (page 96)
  Informs the receiver that one or more touches has moved.
- `touchesCancelledWithEvent:` (page 95)
  Informs the receiver that tracking of touches has been cancelled for any reason..
- `touchesEndedWithEvent:` (page 95)
  Returns that a set of touches have been removed.
- `wantsForwardedScrollEventsForAxis:` (page 101)
  Returns whether to forward elastic scrolling gesture events up the responder.

Setting the Writing Direction

- `makeBaseWritingDirectionLeftToRight:` (page 44)
  Sets the paragraph base writing direction to be left to right.
- `makeBaseWritingDirectionNatural:` (page 45)
  Sets the paragraph base writing direction to be natural.
- `makeBaseWritingDirectionRightToLeft:` (page 46)
  Sets the paragraph base writing direction to be right to left.
- `makeTextWritingDirectionLeftToRight:` (page 46)
  Sets the character level attributed string direction attribute for left to right text.
- `makeTextWritingDirectionNatural:` (page 47)
  Removes the character-level writing direction attribute
- `makeTextWritingDirectionRightToLeft:` (page 47)
  Sets the character-level writing direction attribute to a single right-to-left embedding.

© 2011 Apple Inc. All Rights Reserved.

Performing Text Find Actions

– `performTextFinderAction:` (page 76)

Performs all find oriented actions.

Class Methods

restorableStateKeyPaths

Returns an array of key paths representing the restorable attributes of the responder.

```
+ (NSArray *)restorableStateKeyPaths
```

Return Value

An array of `NSString` objects, each of which contains a key path to one of the responder's attributes.

Discussion

You can use this method instead of, or in addition to, the `encodeRestorableStateWithCoder:` (page 33) and `restoreStateWithCoder:` (page 80) methods to save and restore the state of your responder. The key paths you return must refer to attributes that are key-value coding and key-value observing compliant.

When changes are detected, the specified attributes are automatically written to disk with the rest of the application's interface-related state. At launch time, the attributes are automatically restored to their previous values.

Availability

Available in OS X v10.7 and later.

Related Sample Code
FullScreenWindow

Declared in

`NSWindowRestoration.h`

© 2011 Apple Inc. All Rights Reserved.

Instance Methods

acceptsFirstResponder

Overridden by subclasses to return YES if the receiver accepts first responder status.

```
- (BOOL)acceptsFirstResponder
```

Discussion

As first responder, the receiver is the first object in the responder chain to be sent key events and action messages. The NSResponder implementation returns NO, indicating that by default a responder object doesn't agree to become first responder.

Availability

Available in OS X v10.0 and later.

See Also

– becomeFirstResponder (page 22)
– resignFirstResponder (page 79)
needsPanelToBecomeKey (NSView)

Related Sample Code

BlastApp
FilterDemo
GLUT
HeightArray
OpenCL NBody Simulation Example

Declared in

NSResponder.h

becomeFirstResponder

Notifies the receiver that it's about to become first responder in its NSWindow.

```
- (BOOL)becomeFirstResponder
```

Discussion

The default implementation returns YES, accepting first responder status. Subclasses can override this method to update state or perform some action such as highlighting the selection, or to return NO, refusing first responder status.

© 2011 Apple Inc. All Rights Reserved.

Use the `NSWindow makeFirstResponder:` method, not this method, to make an object the first responder. Never invoke this method directly.

Availability

Available in OS X v10.0 and later.

See Also

– `resignFirstResponder` (page 79)
– `acceptsFirstResponder` (page 22)

Related Sample Code

Dicey
FilterDemo
From A View to A Movie
GLUT
HeightArray

Declared in

`NSResponder.h`

beginGestureWithEvent:

Informs the receiver that the user has begun a touch gesture.

```
– (void)beginGestureWithEvent:(NSEvent *)event
```

Parameters

`event`

An event object representing the gesture beginning.

Discussion

The event will be sent to the view under the touch in the key window.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

cancelOperation:

Implemented by subclasses to cancel the current operation.

© 2011 Apple Inc. All Rights Reserved.

```
- (void)cancelOperation:(id)sender
```

Parameters

`sender`

The object invoking this method.

Discussion

This method is bound to the Escape and Command-. (period) keys. The key window first searches the view hierarchy for a view whose key equivalent is Escape or Command-., whichever was entered. If none of these views handles the key equivalent, the window sends a default action message of `cancelOperation:` to the first responder and from there the message travels up the responder chain.

If no responder in the responder chain implements `cancelOperation:`, the key window searches the view hierarchy for a view whose key equivalent is Escape (note that this may be redundant if the original key equivalent was Escape). If no such responder is found, then a `cancel:` action message is sent to the first responder in the responder chain that implements it.

`NSResponder` declares but does not implement this method.

Availability

Available in OS X v10.3 and later.

Declared in

`NSResponder.h`

capitalizeWord:

Implemented by subclasses to capitalize the word or words surrounding the insertion point or selection, expanding the selection if necessary.

```
- (void)capitalizeWord:(id)sender
```

Parameters

`sender`

The object invoking the method.

Discussion

If either end of the selection partially covers a word, that entire word is made lowercase. The `sender` argument is typically the object that invoked this method. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

© 2011 Apple Inc. All Rights Reserved.

See Also

– lowercaseWord: (page 43)
– uppercaseWord: (page 98)
– changeCaseOfLetter: (page 25)

Declared in

NSResponder.h

centerSelectionInVisibleArea:

Implemented by subclasses to scroll the selection, whatever it is, inside its visible area.

```
– (void)centerSelectionInVisibleArea:(id)sender
```

Parameters

sender

The object that invoked the method (typically).

Discussion

NSResponder declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– scrollLineDown: (page 82)
– scrollLineUp: (page 83)
– scrollPageDown: (page 83)
– scrollPageUp: (page 84)

Declared in

NSResponder.h

changeCaseOfLetter:

Implemented by subclasses to change the case of a letter or letters in the selection, perhaps by opening a panel with capitalization options or by cycling through possible case combinations.

```
– (void)changeCaseOfLetter:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

© 2011 Apple Inc. All Rights Reserved.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `lowercaseWord:` (page 43)
– `uppercaseWord:` (page 98)
– `capitalizeWord:` (page 24)

Declared in

`NSResponder.h`

complete:

Implemented by subclasses to complete an operation in progress or a partially constructed element.

```
– (void)complete:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

This method can be interpreted, for example, as a request to attempt expansion of a partial word, such as for expanding a glossary shortcut, or to close a graphics item being drawn. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

cursorUpdate:

Informs the receiver that the mouse cursor has moved into a cursor rectangle.

```
– (void)cursorUpdate:(NSEvent *)event
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`event`

An object encapsulating information about the cursor-update event (`NSCursorUpdate`).

Discussion

Override this method to set the cursor image. The default implementation uses cursor rectangles, if cursor rectangles are currently valid. If they are not, it calls `super` to send the message up the responder chain.

If the responder implements this method, but decides not to handle a particular event, it should invoke the superclass implementation of this method.

Availability

Available in OS X v10.5 and later.

Declared in

`NSResponder.h`

deleteBackward:

Implemented by subclasses to delete the selection, if there is one, or a single element backward from the insertion point (a letter or character in text, for example).

```
- (void)deleteBackward:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

deleteBackwardByDecomposingPreviousCharacter:

Implemented by subclasses to delete the selection, if there is one, or a single character backward from the insertion point.

```
- (void)deleteBackwardByDecomposingPreviousCharacter:(id)sender
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If the previous character is canonically decomposable, this method should try to delete only the last character in the grapheme cluster (for example, deleting "a"+ "´" results in "a"). `NSResponder` declares but does not implement this method.

Availability

Available in OS X v10.3 and later.

Declared in

`NSResponder.h`

deleteForward:

Implemented by subclasses to delete the selection, if there is one, or a single element forward from the insertion point (a letter or character in text, for example).

```
- (void)deleteForward:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

deleteToBeginningOfLine:

Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the beginning of a line (typically of text).

```
- (void)deleteToBeginningOfLine:(id)sender
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`sender`

Typically the object that invoked this method.

Discussion

Also places the deleted text into the kill buffer. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `yank:` (page 102)

Declared in

`NSResponder.h`

deleteToBeginningOfParagraph:

Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the beginning of a paragraph of text.

```
– (void)deleteToBeginningOfParagraph:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

Also places the deleted text into the kill buffer. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `yank:` (page 102)

Declared in

`NSResponder.h`

deleteToEndOfLine:

Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the end of a line (typically of text).

© 2011 Apple Inc. All Rights Reserved.

```
- (void)deleteToEndOfLine:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

Also places the deleted text into the kill buffer. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

deleteToEndOfParagraph:

Implemented by subclasses to delete the selection, if there is one, or all text from the insertion point to the end of a paragraph of text.

```
- (void)deleteToEndOfParagraph:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

Also places the deleted text into the kill buffer. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `yank:` (page 102)

Declared in

`NSResponder.h`

deleteToMark:

Implemented by subclasses to delete the selection, if there is one, or all items from the insertion point to a previously placed mark, including the selection itself if not empty.

© 2011 Apple Inc. All Rights Reserved.

```
- (void)deleteToMark:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

Also places the deleted text into the kill buffer. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `setMark:` (page 88)
– `selectToMark:` (page 87)
– `yank:` (page 102)

Declared in

`NSResponder.h`

deleteWordBackward:

Implemented by subclasses to delete the selection, if there is one, or a single word backward from the insertion point.

```
- (void)deleteWordBackward:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

deleteWordForward:

Implemented by subclasses to delete the selection, if there is one, or a single word forward from the insertion point.

```
- (void)deleteWordForward:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

doCommandBySelector:

Attempts to perform the indicated command.

```
- (void)doCommandBySelector:(SEL)aSelector
```

Parameters

`aSelector`

The selector identifying the method.

Discussion

If the receiver responds to `aSelector`, it invokes the method with `nil` as the argument. If the receiver doesn't respond, it sends this message to its next responder with the same selector. `NSWindow` and `NSApplication` also send the message to their delegates. If the receiver has no next responder or delegate, it beeps.

Availability

Available in OS X v10.0 and later.

See Also

`- tryToPerform:with:` (page 97)
`sendAction:to:from:` (NSApplication)

© 2011 Apple Inc. All Rights Reserved.

Declared in
`NSResponder.h`

encodeRestorableStateWithCoder:

Saves the interface-related state of the responder.

```
- (void)encodeRestorableStateWithCoder:(NSCoder *)coder
```

Parameters
`coder`
The coder object in which to save the responder's interface-related state.

Discussion
This method is part of the window restoration system and is called at appropriate times to save the visual state of your responder to the specified archive. The default implementation of this method does nothing but specific subclasses (such as `NSView` and `NSWindow`) override it to save important state information. Therefore, if you override this method, you should always call `super` at some point in your implementation.

Subclasses can override this method and use it to restore any information that would be needed to restore the responder to its current state. For example, the `NSTabView` class uses this method to save information about the currently selected tab. You must store enough data to reconfigure the responder and return it to its current state during a subsequent launch of the application.

For information about using a coder object to write data to an archive, see *Archives and Serializations Programming Guide*.

Availability
Available in OS X v10.7 and later.

See Also
`+ restorableStateKeyPaths` (page 21)

Declared in
`NSWindowRestoration.h`

endGestureWithEvent:

Informs the receiver that the user has ended a touch gesture.

```
- (void)endGestureWithEvent:(NSEvent *)event
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`event`

An event object representing the gesture end.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

flagsChanged:

Informs the receiver that the user has pressed or released a modifier key (Shift, Control, and so on).

```
- (void)flagsChanged:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the modifier-key event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

flushBufferedKeyEvents

Overridden by subclasses to clear any unprocessed key events.

```
- (void)flushBufferedKeyEvents
```

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

helpRequested:

Displays context-sensitive help for the receiver if such exists; otherwise passes this message to the next responder.

```
- (void)helpRequested:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the help-request event.

Discussion

`NSWindow` invokes this method automatically when the user clicks for help—while processing `theEvent`. Subclasses need not override this method, and application code shouldn't directly invoke it.

Availability

Available in OS X v10.0 and later.

See Also

– `showContextHelp:` (page 91)

Declared in

`NSResponder.h`

indent:

Implemented by subclasses to indent the selection or the insertion point if there is no selection.

```
- (void)indent:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

insertBacktab:

Implemented by subclasses to handle a backward tab.

```
- (void)insertBacktab:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

A field editor might respond to this method by selecting the field before it, while a regular text object either doesn't respond to or ignores such a message. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

insertContainerBreak:

Implemented by subclasses to insert a container break (typically a page break) at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertContainerBreak:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method. `NSTextView` implements it to insert an `NSFormFeedCharacter` character (0x000c).

Availability

Available in OS X v10.4 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

insertDoubleQuoteIgnoringSubstitution:

Implemented by subclasses to insert a double quote character at the insertion point without interference by automatic quote correction.

```
- (void)insertDoubleQuoteIgnoringSubstitution:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

insertLineBreak:

Implemented by subclasses to insert a line break (as distinguished from a paragraph break) at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertLineBreak:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method. `NSTextView` implements it to insert an `NSLineSeparatorCharacter` character (0x2028).

Availability

Available in OS X v10.4 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

insertNewline:

Implemented by subclasses to insert a newline character at the insertion point or selection, deleting the selection if there is one, or to end editing if the receiver is a text field or other field editor.

```
- (void)insertNewline:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

insertNewlineIgnoringFieldEditor:

Implemented by subclasses to insert a line-break character at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertNewlineIgnoringFieldEditor:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

Unlike `insertNewline:` (page 38), this method always inserts a line-break character and doesn't cause the receiver to end editing. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

insertParagraphSeparator:

Implemented by subclasses to insert a paragraph separator at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertParagraphSeparator:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

insertSingleQuoteIgnoringSubstitution:

Implemented by subclasses to insert a single quote character at the insertion point without interference by automatic quote correction.

```
- (void)insertSingleQuoteIgnoringSubstitution:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

insertTab:

Implemented by subclasses to insert a tab character at the insertion point or selection, deleting the selection if there is one, or to end editing if the receiver is a text field or other field editor.

```
- (void)insertTab:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

insertTabIgnoringFieldEditor:

Implemented by subclasses to insert a tab character at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertTabIgnoringFieldEditor:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

Unlike `insertTab:` (page 40), this method always inserts a tab character and doesn't cause the receiver to end editing. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

insertText:

Overridden by subclasses to insert the supplied string at the insertion point or selection, deleting the selection if there is one.

```
- (void)insertText:(id)aString
```

Parameters

`aString`

The string to insert or replace the selection with. `aString` can be either an `NSString` object or an `NSAttributedString` object.

Discussion

This method is often invoked by the system input manager after the receiver sends a interpretKeyEvents: (page 41) message. The `NSResponder` implementation simply passes this message to the next responder, or beeps if there is no next responder.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

interpretKeyEvents:

Invoked by subclasses from their `keyDown:` (page 42) method to handle a series of key events.

```
- (void)interpretKeyEvents:(NSArray *)eventArray
```

Parameters

`eventArray`

An array of key-event characters to give to the system input manager.

Discussion

This method sends the character input in `eventArray` to the system input manager for interpretation as text to insert or commands to perform. The input manager responds to the request by sending `insertText:` (page 41) and `doCommandBySelector:` (page 32) messages back to the invoker of this method. Subclasses shouldn't override this method.

See the `NSInputManager` and `NSTextInput` class and protocol specifications for more information on input management.

© 2011 Apple Inc. All Rights Reserved.

Availability
Available in OS X v10.0 and later.

Declared in
`NSResponder.h`

invalidateRestorableState

Marks the responder's interface-related state as dirty.

```
- (void)invalidateRestorableState
```

Discussion
Call this method whenever the restorable state of your responder changes. This method marks the responder's state as dirty, which causes that state to be written to disk at some point in the future. Do not override this method.

Availability
Available in OS X v10.7 and later.

See Also
– `encodeRestorableStateWithCoder:` (page 33)
+ `restorableStateKeyPaths` (page 21)

Declared in
`NSWindowRestoration.h`

keyDown:

Informs the receiver that the user has pressed a key.

```
- (void)keyDown:(NSEvent *)theEvent
```

Parameters
`theEvent`

An object encapsulating information about the key-down event.

Discussion
The receiver can interpret `theEvent` itself, or pass it to the system input manager using `interpretKeyEvents:` (page 41). The default implementation simply passes this message to the next responder.

© 2011 Apple Inc. All Rights Reserved.

Availability

Available in OS X v10.0 and later.

Related Sample Code

DragItemAround

GLFullScreen

Declared in

`NSResponder.h`

keyUp:

Informs the receiver that the user has released a key.

```
- (void)keyUp:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the key-up event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

lowercaseWord:

Implemented by subclasses to make lowercase every letter in the word or words surrounding the insertion point or selection, expanding the selection if necessary.

```
- (void)lowercaseWord:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If either end of the selection partially covers a word, that entire word is made lowercase. `NSResponder` declares, but doesn't implement this method.

© 2011 Apple Inc. All Rights Reserved.

Availability

Available in OS X v10.0 and later.

See Also

– uppercaseWord: (page 98)
– capitalizeWord: (page 24)
– changeCaseOfLetter: (page 25)

Declared in

NSResponder.h

magnifyWithEvent:

Informs the receiver that the user has begun a pinch gesture.

```
– (void)magnifyWithEvent:(NSEvent *)event
```

Parameters

event

An event object representing the magnify gesture.

Discussion

The event will be sent to the view under the touch in the key window.

Availability

Available in OS X v10.6 and later.

Declared in

NSResponder.h

makeBaseWritingDirectionLeftToRight:

Sets the paragraph base writing direction to be left to right.

```
– (void)makeBaseWritingDirectionLeftToRight:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

© 2011 Apple Inc. All Rights Reserved.

Discussion
Sets the `NSAttributedString` key `NSWritingDirectionAttributeName` to `NSWritingDirectionLeftToRight`.

This action method is intended to be used both as the target of a menu item and for key bindings. The base writing direction methods should be the target of three menu items in a submenu, under the Edit menu.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability
Available in OS X v10.6 and later.

Declared in
`NSResponder.h`

makeBaseWritingDirectionNatural:

Sets the paragraph base writing direction to be natural.

```
- (void)makeBaseWritingDirectionNatural:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
Natural directionality is determined from the text in accordance with the Unicode bi-di algorithm. For more information see `NSParagraphStyle`.

Sets the `NSAttributedString`key `NSWritingDirectionAttributeName` to `NSTextWritingDirectionEmbedding`.

This action method is intended to be used both as the target of a menu item and for key bindings. The base writing direction methods should be the target of three menu items in a submenu, under the Edit menu.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability
Available in OS X v10.6 and later.

Declared in
`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

makeBaseWritingDirectionRightToLeft:

Sets the paragraph base writing direction to be right to left.

```
- (void)makeBaseWritingDirectionRightToLeft:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

Sets the `NSAttributedString` key `NSWritingDirectionAttributeName` to `NSWritingDirectionRightToLeft`.

This action method is intended to be used both as the target of a menu item and for key bindings. The base writing direction methods should be the target of three menu items in a submenu, under the Edit menu.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

makeTextWritingDirectionLeftToRight:

Sets the character level attributed string direction attribute for left to right text.

```
- (void)makeTextWritingDirectionLeftToRight:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

Sets the `NSAttributedString` `NSWritingDirectionAttributeName` to `NSWritingDirectionLeftToRight`.

This action method is intended to be used both as the target of a menu item and for key bindings. The text writing directions should be the target of three similar menu items in a submenu under the Edit menu.

© 2011 Apple Inc. All Rights Reserved.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability
Available in OS X v10.6 and later.

Declared in
`NSResponder.h`

makeTextWritingDirectionNatural:

Removes the character-level writing direction attribute

```
- (void)makeTextWritingDirectionNatural:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
Removes the `NSWritingDirectionAttributeName` from an `NSAttributedString`.

This action method is intended to be used both as the target of a menu item and for key bindings. The text writing directions should be the target of three similar menu items in a submenu under the Edit menu.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability
Available in OS X v10.6 and later.

Declared in
`NSResponder.h`

makeTextWritingDirectionRightToLeft:

Sets the character-level writing direction attribute to a single right-to-left embedding.

```
- (void)makeTextWritingDirectionRightToLeft:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

© 2011 Apple Inc. All Rights Reserved.

Discussion

Sets the `NSAttributedString` key `NSWritingDirectionAttributeName` to `NSWritingDirectionRightToLeft`.

This action method is intended to be used both as the target of a menu item and for key bindings. The text writing directions should be the target of three similar menu items in a submenu under the Edit menu.

Default key bindings will also be provided for this method but will only be enabled for users of Hebrew or Arabic, or those who have otherwise enabled a suitable preference.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

menu

Returns the receiver's menu.

```
- (NSMenu *)menu
```

Discussion

For `NSApplication` this menu is the same as the menu returned by its `mainMenu` method.

Availability

Available in OS X v10.0 and later.

See Also

`- setMenu:` (page 89)
`menuForEvent:` (NSView)
`defaultMenu` (NSView)

Related Sample Code

ButtonMadness
GLUT
GridMenu
MenuItemView
UIElementInspector

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

mouseDown:

Informs the receiver that the user has pressed the left mouse button.

```
- (void)mouseDown:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the mouse-down event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

Related Sample Code

GLFullScreen

Declared in

`NSResponder.h`

mouseDragged:

Informs the receiver that the user has moved the mouse with the left button pressed.

```
- (void)mouseDragged:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the mouse-dragged event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

Related Sample Code

GLFullScreen

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

mouseEntered:

Informs the receiver that the cursor has entered a tracking rectangle.

```
- (void)mouseEntered:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the mouse-entered event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

mouseExited:

Informs the receiver that the cursor has exited a tracking rectangle.

```
- (void)mouseExited:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the mouse-exited event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

mouseMoved:

Informs the receiver that the mouse has moved.

```
- (void)mouseMoved:(NSEvent *)theEvent
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`theEvent`

An object encapsulating information about the mouse-moved event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

See Also

`setAcceptsMouseMovedEvents:` (NSWindow)

Declared in

`NSResponder.h`

mouseUp:

Informs the receiver that the user has released the left mouse button.

```
- (void)mouseUp:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the mouse-up event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

Related Sample Code

GLFullScreen

Declared in

`NSResponder.h`

moveBackward:

Implemented by subclasses to move the selection or insertion point one element or character backward.

```
- (void)moveBackward:(id)sender
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`sender`

Typically the object that invoked this method.

Discussion

In text, if there is a selection it should be deselected, and the insertion point should be placed at the beginning of the former selection. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

moveBackwardAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection backward by one element or character.

```
- (void)moveBackwardAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If the end being modified is the backward end, this method expands the selection; if the end being modified is the forward end, it reduces the selection. The first `moveBackwardAndModifySelection:` or `moveForwardAndModifySelection:` (page 54) method in a series determines the end being modified by always expanding. Hence, this method results in the backward end becoming the mobile one if invoked first. By default, `moveLeftAndModifySelection:` (page 55) is bound to the left arrow key.

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

moveDown:

Implemented by subclasses to move the selection or insertion point one element or character down.

© 2011 Apple Inc. All Rights Reserved.

```
- (void)moveDown:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

In text, if there is a selection it should be deselected, and the insertion point should be placed below the beginning of the former selection. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

moveDownAndModifySelection:

Implemented by subclasses to expand or reduce the top or bottom end of the selection downward by one element, character, or line (whichever is appropriate for text direction).

```
- (void)moveDownAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If the end being modified is the bottom, this method expands the selection; if the end being modified is the top, it reduces the selection. The first `moveDownAndModifySelection:` or `moveUpAndModifySelection:` (page 66) method in a series determines the end being modified by always expanding. Hence, this method results in the bottom end becoming the mobile one if invoked first.

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

moveForward:

Implemented by subclasses to move the selection or insertion point one element or character forward.

```
- (void)moveForward:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

In text, if there is a selection it should be deselected, and the insertion point should be placed at the end of the former selection. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

moveForwardAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection forward by one element or character.

```
- (void)moveForwardAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If the end being modified is the backward end, this method reduces the selection; if the end being modified is the forward end, it expands the selection. The first `moveBackwardAndModifySelection:` (page 52) or `moveForwardAndModifySelection:` method in a series determines the end being modified by always expanding. Hence, this method results in the forward end becoming the mobile one if invoked first. By default, `moveRightAndModifySelection:` (page 57) is bound to the right arrow key.

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

© 2011 Apple Inc. All Rights Reserved.

Declared in
`NSResponder.h`

moveLeft:

Implemented by subclasses to move the selection or insertion point one element or character to the left.

```
- (void)moveLeft:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
In text, if there is a selection it should be deselected, and the insertion point should be placed at the left end of the former selection. `NSResponder` declares but doesn't implement this method.

Availability
Available in OS X v10.0 and later.

Declared in
`NSResponder.h`

moveLeftAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection to the left (display order) by one element or character.

```
- (void)moveLeftAndModifySelection:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
If the end being modified is the left end, this method expands the selection; if the end being modified is the right end, it reduces the selection. The first `moveLeftAndModifySelection:` or `moveRightAndModifySelection:` method in a series determines the end being modified by always expanding. Hence, this method results in the left end becoming the mobile one if invoked first. By default, this method is bound to the left arrow key.

`NSResponder` declares but doesn't implement this method.

© 2011 Apple Inc. All Rights Reserved.

The essential difference between this method and the corresponding `moveBackwardAndModifySelection:` is that the latter method moves in logical order, which can differ in bidirectional text, whereas this method moves in display order.

Availability
Available in OS X v10.3 and later.

Declared in
`NSResponder.h`

moveParagraphBackwardAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the beginning of the previous paragraph, expanding or reducing the current selection.

```
- (void)moveParagraphBackwardAndModifySelection:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
If the cursor is already at the beginning of a paragraph, the selection moves backward to the beginning of the previous paragraph.

`NSResponder` declares but doesn't implement this method.

Availability
Available in OS X v10.6 and later.

Declared in
`NSResponder.h`

moveParagraphForwardAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the beginning of the next paragraph, expanding or reducing the current selection.

```
- (void)moveParagraphForwardAndModifySelection:(id)sender
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If the cursor is already at the end of a paragraph, the selection moves forward to the end of the next paragraph.

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

moveRight:

Implemented by subclasses to move the selection or insertion point one element or character to the right.

```
- (void)moveRight:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

In text, if there is a selection it should be deselected, and the insertion point should be placed at the right end of the former selection. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

moveRightAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection to the right (display order) by one element or character.

```
- (void)moveRightAndModifySelection:(id)sender
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If the end being modified is the left end, this method reduces the selection; if the end being modified is the right end, it expands the selection. The first `moveLeftAndModifySelection:` (page 55) or `moveRightAndModifySelection:` method in a series determines the end being modified by always expanding. Hence, this method results in the right end becoming the mobile one if invoked first. By default, this method is bound to the right arrow key.

`NSResponder` declares but doesn't implement this method.

The essential difference between this method and the corresponding `moveForwardAndModifySelection:` (page 54) is that the latter method moves in logical order, which can differ in bidirectional text, whereas this method moves in display order.

Availability

Available in OS X v10.3 and later.

Declared in

`NSResponder.h`

moveToBeginningOfDocument:

Implemented by subclasses to move the selection to the first element of the document or the insertion point to the beginning.

```
- (void)moveToBeginningOfDocument:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

moveToBeginningOfDocumentAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the beginning of the document, expanding or reducing the current selection.

```
- (void)moveToBeginningOfDocumentAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

moveToBeginningOfLine:

Implemented by subclasses to move the selection to the first element of the selected line or the insertion point to the beginning of the line.

```
- (void)moveToBeginningOfLine:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

moveToBeginningOfLineAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the beginning of the line, expanding or reducing the current selection.

```
- (void)moveToBeginningOfLineAndModifySelection:(id)sender
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

moveToBeginningOfParagraph:

Implemented by subclasses to move the insertion point to the beginning of the selected paragraph.

```
- (void)moveToBeginningOfParagraph:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

moveToBeginningOfParagraphAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the beginning of the current paragraph, expanding or reducing the current selection.

```
- (void)moveToBeginningOfParagraphAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

© 2011 Apple Inc. All Rights Reserved.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

moveToEndOfDocument:

Implemented by subclasses to move the selection to the last element of the document or the insertion point to the end.

```
- (void)moveToEndOfDocument:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

moveToEndOfDocumentAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the end of the document, expanding or reducing the current selection.

```
- (void)moveToEndOfDocumentAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Availability

Available in OS X v10.6 and later.

© 2011 Apple Inc. All Rights Reserved.

Declared in
`NSResponder.h`

moveToEndOfLine:

Implemented by subclasses to move the selection to the last element of the selected line or the insertion point to the end of the line.

```
- (void)moveToEndOfLine:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in OS X v10.0 and later.

Declared in
`NSResponder.h`

moveToEndOfLineAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the end of the line, expanding or reducing the current selection.

```
- (void)moveToEndOfLineAndModifySelection:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in OS X v10.6 and later.

Declared in
`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

moveToEndOfParagraph:

Implemented by subclasses to move the insertion point to the end of the selected paragraph.

```
- (void)moveToEndOfParagraph:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

moveToEndOfParagraphAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the end of the current paragraph, expanding or reducing the current selection.

```
- (void)moveToEndOfParagraphAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

moveToLeftEndOfLine:

Implemented by subclasses to move the selection or insertion point to the left end of the line.

© 2011 Apple Inc. All Rights Reserved.

```
- (void)moveToLeftEndOfLine:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

In text, if there is a selection it should be deselected, and the insertion point should be placed at left end of the line. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

moveToLeftEndOfLineAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the left end of the line, expanding or contracting the selection as required.

```
- (void)moveToLeftEndOfLineAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

moveToRightEndOfLine:

Implemented by subclasses to move the selection or insertion point to the right end of the line

```
- (void)moveToRightEndOfLine:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

© 2011 Apple Inc. All Rights Reserved.

Availability
Available in OS X v10.6 and later.

Declared in
`NSResponder.h`

moveToRightEndOfLineAndModifySelection:

Implemented by subclasses to move the selection or insertion point to the right end of the line, expanding or contracting the selection as required.

```
- (void)moveToRightEndOfLineAndModifySelection:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Availability
Available in OS X v10.6 and later.

Declared in
`NSResponder.h`

moveUp:

Implemented by subclasses to move the selection or insertion point one element or character up.

```
- (void)moveUp:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
In text, if there is a selection it should be deselected, and the insertion point should be placed above the beginning of the former selection. `NSResponder` declares but doesn't implement this method.

Availability
Available in OS X v10.0 and later.

Declared in
`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

moveUpAndModifySelection:

Implemented by subclasses to expand or reduce the top or bottom end of the selection upward by one element, character, or line (whichever is appropriate for text direction).

```
- (void)moveUpAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If the end being modified is the bottom, this method reduces the selection; if the end being modified is the top, it expands the selection. The first `moveDownAndModifySelection:` (page 53) or `moveUpAndModifySelection:` method in a series determines the end being modified by always expanding. Hence, this method results in the top end becoming the mobile one if invoked first.

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

moveWordBackward:

Implemented by subclasses to move the selection or insertion point one word backward.

```
- (void)moveWordBackward:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If there is a selection it should be deselected, and the insertion point should be placed at the end of the first word preceding the former selection. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

© 2011 Apple Inc. All Rights Reserved.

See Also

– `moveWordLeft:` (page 68)

Declared in

`NSResponder.h`

moveWordBackwardAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection backward by one whole word.

```
– (void)moveWordBackwardAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If the end being modified is the backward end, this method expands the selection; if the end being modified is the forward end, it reduces the selection. The first `moveWordBackwardAndModifySelection:` or `moveWordForwardAndModifySelection:` (page 68) method in a series determines the end being modified by always expanding. Hence, this method results in the backward end becoming the mobile one if invoked first.

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `moveWordLeftAndModifySelection:` (page 69)

Declared in

`NSResponder.h`

moveWordForward:

Implemented by subclasses to move the selection or insertion point one word forward, in logical order.

```
– (void)moveWordForward:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

© 2011 Apple Inc. All Rights Reserved.

Discussion

If there is a selection it should be deselected, and the insertion point should be placed at the beginning of the first word following the former selection. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `moveWordRight:` (page 70)

Declared in

`NSResponder.h`

moveWordForwardAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection forward by one whole word.

```
– (void)moveWordForwardAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If the end being modified is the backward end, this method reduces the selection; if the end being modified is the forward end, it expands the selection. The first `moveWordBackwardAndModifySelection:` (page 67) or `moveWordForwardAndModifySelection:` method in a series determines the end being modified by always expanding. Hence, this method results in the forward end becoming the mobile one if invoked first. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `moveWordRightAndModifySelection:` (page 70)

Declared in

`NSResponder.h`

moveWordLeft:

Implemented by subclasses to move the selection or insertion point one word to the left, in display order.

© 2011 Apple Inc. All Rights Reserved.

```
- (void)moveWordLeft:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If there is a selection it should be deselected, and the insertion point should be placed at the end of the first word to the left of the former selection. `NSResponder` declares but doesn't implement this method.

The main difference between this method and the corresponding `moveWordBackward:` (page 66) method is that the latter moves in logical order, which is important in bidirectional text, whereas this method moves in display order.

Availability

Available in OS X v10.3 and later.

Declared in

`NSResponder.h`

moveWordLeftAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection left by one whole word in display order.

```
- (void)moveWordLeftAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If the end being modified is the left end, this method expands the selection; if the end being modified is the right end, it reduces the selection. The first `moveWordLeftAndModifySelection:` or `moveWordRightAndModifySelection:` (page 70) method in a series determines the end being modified by always expanding. Hence, this method results in the left end becoming the mobile one if invoked first.

`NSResponder` declares but doesn't implement this method.

The main difference between this method and the corresponding `moveWordBackwardAndModifySelection:` (page 67) method is that the latter moves in logical order, which is important in bidirectional text, whereas this method moves in display order.

© 2011 Apple Inc. All Rights Reserved.

Availability
Available in OS X v10.3 and later.

Declared in
`NSResponder.h`

moveWordRight:

Implemented by subclasses to move the selection or insertion point one word right.

```
- (void)moveWordRight:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
If there is a selection it should be deselected, and the insertion point should be placed at the beginning of the first word to the right of the former selection. `NSResponder` declares but doesn't implement this method.

The main difference between this method and the corresponding `moveWordForward:` (page 67) method is that the latter moves in logical order, which is important in bidirectional text, whereas this method moves in display order.

Availability
Available in OS X v10.3 and later.

Declared in
`NSResponder.h`

moveWordRightAndModifySelection:

Implemented by subclasses to expand or reduce either end of the selection to the right by one whole word.

```
- (void)moveWordRightAndModifySelection:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

© 2011 Apple Inc. All Rights Reserved.

Discussion

If the end being modified is the backward end, this method reduces the selection; if the end being modified is the forward end, it expands the selection. The first `moveWordBackwardAndModifySelection:` (page 67) or `moveWordForwardAndModifySelection:` method in a series determines the end being modified by always expanding. Hence, this method results in the forward end becoming the mobile one if invoked first. `NSResponder` declares but doesn't implement this method.

The main difference between this method and the corresponding `moveWordForwardAndModifySelection:` (page 68) method is that the latter moves in logical order, which is important in bidirectional text, whereas this method moves in display order.

Availability

Available in OS X v10.3 and later.

Declared in

`NSResponder.h`

nextResponder

Returns the receiver's next responder, or `nil` if it has none.

```
- (NSResponder *)nextResponder
```

Availability

Available in OS X v10.0 and later.

See Also

- `setNextResponder:` (page 89)
- `noResponderFor:` (page 71)

Declared in

`NSResponder.h`

noResponderFor:

Handles the case where an event or action message falls off the end of the responder chain.

```
- (void)noResponderFor:(SEL)eventSelector
```

Parameters

`eventSelector`

A selector identifying the action or event message.

© 2011 Apple Inc. All Rights Reserved.

Discussion

The default implementation beeps if `eventSelector` is `keyDown:` (page 42).

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

otherMouseDown:

Informs the receiver that the user has pressed a mouse button other than the left or right one.

```
- (void)otherMouseDown:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the mouse-down event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.1 and later.

Declared in

`NSResponder.h`

otherMouseDragged:

Informs the receiver that the user has moved the mouse with a button other than the left or right button pressed.

```
- (void)otherMouseDragged:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the mouse-dragged event.

Discussion

The default implementation simply passes this message to the next responder.

© 2011 Apple Inc. All Rights Reserved.

Availability
Available in OS X v10.1 and later.

Declared in
`NSResponder.h`

otherMouseUp:

Informs the receiver that the user has released a mouse button other than the left or right button.

```
- (void)otherMouseUp:(NSEvent *)theEvent
```

Parameters
`theEvent`
An object encapsulating information about the mouse-up event.

Discussion
The default implementation simply passes this message to the next responder.

Availability
Available in OS X v10.1 and later.

Declared in
`NSResponder.h`

pageDown:

Implemented by subclasses to scroll the receiver down (or back) one page in its scroll view, also moving the insertion point to the top of the newly displayed page.

```
- (void)pageDown:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in OS X v10.0 and later.

© 2011 Apple Inc. All Rights Reserved.

See Also

– scrollPageDown: (page 83)
– scrollPageUp: (page 84)

Declared in

NSResponder.h

pageDownAndModifySelection:

Implemented by subclasses to scroll the receiver down (or back) one page in its scroll view, also moving the insertion point to the top of the newly displayed page. The selection is expanded or contracted as required.

```
– (void)pageDownAndModifySelection:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

NSResponder declares but doesn't implement this method.

Availability

Available in OS X v10.6 and later.

Declared in

NSResponder.h

pageUp:

Implemented by subclasses to scroll the receiver up (or forward) one page in its scroll view, also moving the insertion point to the top of the newly displayed page.

```
– (void)pageUp:(id)sender
```

Parameters

sender

Typically the object that invoked this method.

Discussion

NSResponder declares but doesn't implement this method.

© 2011 Apple Inc. All Rights Reserved.

Availability

Available in OS X v10.0 and later.

See Also

– `scrollPageDown:` (page 83)
– `scrollPageUp:` (page 84)

Declared in

`NSResponder.h`

pageUpAndModifySelection:

Implemented by subclasses to scroll the receiver up (or forward) one page in its scroll view, also moving the insertion point to the top of the newly displayed page. The selection is expanded or contracted as necessary.

```
– (void)pageUpAndModifySelection:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

performKeyEquivalent:

Overridden by subclasses to handle a key equivalent.

```
– (BOOL)performKeyEquivalent:(NSEvent *)theEvent
```

Parameters

`theEvent`

An event object that represents the key equivalent pressed.

Discussion

If the character code or codes in `theEvent` match the receiver's key equivalent, the receiver should respond to the event and return `YES`. The default implementation does nothing and returns `NO`.

© 2011 Apple Inc. All Rights Reserved.

> **Note:** `performKeyEquivalent:` (page 75) takes an `NSEvent` object as its argument, while `performMnemonic:` (page 104) takes an `NSString` object containing the uninterpreted characters of the key event. You should extract the characters for a key equivalent using the `NSEvent` method `charactersIgnoringModifiers`.

Availability
Available in OS X v10.0 and later.

See Also
`performKeyEquivalent:` (NSView)
`performKeyEquivalent:` (NSButton)

Declared in
`NSResponder.h`

performTextFinderAction:

Performs all find oriented actions.

```
- (void)performTextFinderAction:(id)sender
```

Parameters
`sender`
The sender of the find action.

Discussion
When an application performs a find action, it should send this message to the responder chain.

A responder of `performTextFinderAction:` is responsible for creating and owning an instance of `NSTextFinder`. Before any other messages are sent to the `NSTextFinder`, you should set its 'client' property to an object which implements the `NSTextFinderClient` protocol.

> **Note:** Before OS X v10.7, the default action for these menu items was `performFindPanelAction:`. Whenever possible which you should update your implementation to use this new action.

Availability
Available in OS X v10.7 and later.

Declared in
`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

presentError:

Presents an error alert to the user as an application-modal dialog.

```
- (BOOL)presentError:(NSError *)anError
```

Parameters

`anError`

An object containing information about an error.

Discussion

The alert displays information found in the `NSError` object `anError`; this information can include error description, recovery suggestion, failure reason, and button titles (all localized). The method returns `YES` if error recovery succeeded and `NO` otherwise. For error recovery to be attempted, an recovery-attempter object (that is, an object conforming to the `NSErrorRecoveryAttempting` informal protocol) must be associated with `anError`.

The default implementation of this method sends `willPresentError:` (page 101) to `self`. By doing this, `NSResponder` gives subclasses an opportunity to customize error presentation. It then forwards the message, passing any customized error object, to the next responder; if there is no next responder, it passes the error object to `NSApp`, which displays a document-modal error alert. When the user dismisses the alert, any recovery attempter associated with the error object is given a chance to recover from the error. See the class description for the precise route up the responder chain (plus document and controller objects) this message might travel.

It is not recommended that you attempt to override this method. If you wish to customize the error presentation, override `willPresentError:` (page 101) instead.

Availability

Available in OS X v10.4 and later.

See Also

– `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 78)

Related Sample Code

DesktopImage

TextEdit

TextSizingExample

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

presentError:modalForWindow:delegate:didPresentSelector:contextInfo:

Presents an error alert to the user as a document-modal sheet attached to document window.

```
- (void)presentError:(NSError *)error modalForWindow:(NSWindow *)aWindow
delegate:(id)delegate didPresentSelector:(SEL)didPresentSelector contextInfo:(void
 *)contextInfo
```

Parameters

`error`

The object encapsulating information about the error.

`aWindow`

The window object identifying the window owning the document-modal sheet.

`delegate`

The modal delegate for the sheet.

`didPresentSelector`

A selector identifying the message to be sent to the modal delegate. The `didPresentSelector` selector must have the signature:

```
- (void)didPresentErrorWithRecovery:(BOOL)didRecover
   contextInfo:(void *)contextInfo
```

`contextInfo`

Supplemental data to be passed to the modal delegate; can be `NULL`.

Discussion

The information displayed in the alert is extracted from the `NSError` object `error`; it may include a description, recovery suggestion, failure reason, and button titles (all localized). Once the user dismisses the alert and any recovery attempter associated with the error object has had a chance to recover from it, the receiver sends a message identified by `didPresentSelector` to the modal delegate `delegate`. (A recovery attempter is an object that conforms to the `NSErrorRecoveryAttempting` informal protocol.)

The modal delegate implements the method identified by `didPresentSelector` to perform any post-error processing if recovery failed or was not attempted (that is, `didRecover` is `NO`). Any supplemental data is passed to the modal delegate via `contextInfo`.

The default implementation of this method sends `willPresentError:` (page 101) to `self`. By doing this, `NSResponder` gives subclasses an opportunity to customize error presentation. It then forwards the message, passing any customized error, to the next responder or; if there is no next responder, it passes the error object

© 2011 Apple Inc. All Rights Reserved.

to NSApp, which displays a document-modal error alert. When the user dismisses the alert, any recovery attempter associated with the error object is given a chance to recover from the error. See the class description for the precise route up the responder chain (plus document and controller objects) this message might travel.

It is not recommended that you attempt to override this method. If you wish to customize the error presentation, override `willPresentError:` (page 101) instead.

Availability

Available in OS X v10.4 and later.

See Also

– `presentError:` (page 77)

Related Sample Code

ClipboardViewer

Sketch

Sketch+Accessibility

TextEdit

ZipBrowser

Declared in

`NSResponder.h`

resignFirstResponder

Notifies the receiver that it's been asked to relinquish its status as first responder in its window.

```
– (BOOL)resignFirstResponder
```

Discussion

The default implementation returns YES, resigning first responder status. Subclasses can override this method to update state or perform some action such as unhighlighting the selection, or to return NO, refusing to relinquish first responder status.

Use the NSWindow `makeFirstResponder:` method, not this method, to make an object the first responder. Never invoke this method directly.

Availability

Available in OS X v10.0 and later.

See Also

– `becomeFirstResponder` (page 22)
– `acceptsFirstResponder` (page 22)

© 2011 Apple Inc. All Rights Reserved.

Related Sample Code
Dicey
From A View to A Movie
From A View to A Picture
OpenALExample
TextInputView

Declared in
`NSResponder.h`

restoreStateWithCoder:

Restores the interface-related state of the responder.

```
- (void)restoreStateWithCoder:(NSCoder *)coder
```

Parameters
`coder`
The coder object to use to restore the responder's interface-related state.

Discussion
This method is part of the window restoration system and is called at launch time to restore the visual state of your responder object. The default implementation does nothing but specific subclasses (such as `NSView` and `NSWindow`) override it and save important state information. Therefore, if you override this method, you should always call `super` at some point in your implementation.

Subclasses can override this method and use it to restore any information that was saved in the `encodeRestorableStateWithCoder:` (page 33) method. You can also use this method to reconfigure the responder to its previous appearance.

For information about using a coder object to read data from an archive, see *Archives and Serializations Programming Guide*.

Availability
Available in OS X v10.7 and later.

Declared in
`NSWindowRestoration.h`

rightMouseDown:

Informs the receiver that the user has pressed the right mouse button.

© 2011 Apple Inc. All Rights Reserved.

```
- (void)rightMouseDown:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the mouse-down event.

Discussion

The default implementation simply passes this message to the next responder.

> **Note:** The `NSView` implementation of this method does not pass the message up the responder chain, it handles it directly. See `rightMouseDown:` in *NSView Class Reference* for details.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

rightMouseDragged:

Informs the receiver that the user has moved the mouse with the right button pressed .

```
- (void)rightMouseDragged:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the mouse-dragged event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

rightMouseUp:

Informs the receiver that the user has released the right mouse button.

```
- (void)rightMouseUp:(NSEvent *)theEvent
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`theEvent`

An object encapsulating information about the mouse-up event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

rotateWithEvent:

Informs the receiver that the user has begun a rotation gesture.

```
- (void)rotateWithEvent:(NSEvent *)event
```

Parameters

`event`

An event object representing the rotate gesture.

Discussion

The event will be sent to the view under the touch in the key window.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

scrollLineDown:

Implemented by subclasses to scroll the receiver one line down in its scroll view, without changing the selection.

```
- (void)scrollLineDown:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

© 2011 Apple Inc. All Rights Reserved.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

`– scrollLineUp:` (page 83)
`lineScroll` (NSScrollView)

Declared in

`NSResponder.h`

scrollLineUp:

Implemented by subclasses to scroll the receiver one line up in its scroll view, without changing the selection.

```
– (void)scrollLineUp:(id)sender
```

Parameters

`sender`
: Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

`– scrollLineDown:` (page 82)
`lineScroll` (NSScrollView)

Declared in

`NSResponder.h`

scrollPageDown:

Implemented by subclasses to scroll the receiver one page down in its scroll view, without changing the selection.

```
– (void)scrollPageDown:(id)sender
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `pageDown:` (page 73)
– `pageUp:` (page 74)
`pageScroll` (NSScrollView)

Declared in

`NSResponder.h`

scrollPageUp:

Implemented by subclasses to scroll the receiver one page up in its scroll view, without changing the selection.

```
– (void)scrollPageUp:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `pageDown:` (page 73)
– `pageUp:` (page 74)
`pageScroll` (NSScrollView)

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

scrollToBeginningOfDocument:

Implemented by subclasses to scroll the receiver to the beginning of the document, without changing the selection.

```
- (void)scrollToBeginningOfDocument:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

scrollToEndOfDocument:

Implemented by subclasses to scroll the receiver to the end of the document, without changing the selection.

```
- (void)scrollToEndOfDocument:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

scrollWheel:

Informs the receiver that the mouse's scroll wheel has moved.

```
- (void)scrollWheel:(NSEvent *)theEvent
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`theEvent`

An object encapsulating information about the wheel-scrolling event.

Discussion

The default implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

selectAll:

Implemented by subclasses to select all selectable elements.

```
- (void)selectAll:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

selectLine:

Implemented by subclasses to select all elements in the line or lines containing the selection or insertion point.

```
- (void)selectLine:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

© 2011 Apple Inc. All Rights Reserved.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in OS X v10.0 and later.

Declared in
`NSResponder.h`

selectParagraph:

Implemented by subclasses to select all paragraphs containing the selection or insertion point.

```
- (void)selectParagraph:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in OS X v10.0 and later.

Declared in
`NSResponder.h`

selectToMark:

Implemented by subclasses to select all items from the insertion point or selection to a previously placed mark, including the selection itself if not empty.

```
- (void)selectToMark:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

© 2011 Apple Inc. All Rights Reserved.

Availability

Available in OS X v10.0 and later.

See Also

– `setMark:` (page 88)
– `deleteToMark:` (page 30)

Declared in

`NSResponder.h`

selectWord:

Implemented by subclasses to extend the selection to the nearest word boundaries outside it (up to, but not including, word delimiters).

```
– (void)selectWord:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

setMark:

Implemented by subclasses to set a mark at the insertion point or selection, which is used by `deleteToMark:` (page 30) and `selectToMark:` (page 87).

```
– (void)setMark:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

© 2011 Apple Inc. All Rights Reserved.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `swapWithMark:` (page 92)

Declared in

`NSResponder.h`

setMenu:

Sets the receiver's menu.

```
– (void)setMenu:(NSMenu *)aMenu
```

Parameters

`aMenu`

The menu object to set as the receiver's menu.

Discussion

If the receiver is an `NSApplication` object, this method sets the main menu, typically set using `setMainMenu:`.

Availability

Available in OS X v10.0 and later.

See Also

– `menu` (page 48)

Related Sample Code

From A View to A Movie

From A View to A Picture

Declared in

`NSResponder.h`

setNextResponder:

Sets the receiver's next responder.

```
– (void)setNextResponder:(NSResponder *)aResponder
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`aResponder`

An object that inherits, directly or indirectly, from `NSResponder`.

Availability

Available in OS X v10.0 and later.

See Also

– `nextResponder` (page 71)

Declared in

`NSResponder.h`

shouldBeTreatedAsInkEvent:

Returns `YES` if the specified event should be treated as an ink event, `NO` if it should be treated as a mouse event.

```
– (BOOL)shouldBeTreatedAsInkEvent:(NSEvent *)theEvent
```

Parameters

`theEvent`

An event object representing the event to be tested.

Discussion

This method provides the ability to distinguish when a pen-down should start inking versus when a pen-down should be treated as a mouse down event. This allows for a write-anywhere model for pen-based input.

The default implementation in `NSApplication` sends the method to the `NSWindow` object under the pen. If the window is inactive, this method returns `YES`, unless the pen-down is in the window drag region. If the window is active, this method is sent to the `NSView` object under the pen.

The default implementation in `NSView` returns `YES`, and `NSControl` overrides and returns `NO`. This allows write-anywhere over most `NSView` objects, but allows the pen to be used to track in controls and to move windows.

A custom view should override this method to get the correct behavior for a pen-down in the view.

Availability

Available in OS X v10.2 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

showContextHelp:

Implemented by subclasses to invoke the help system, displaying information relevant to the receiver and its current state.

```
- (void)showContextHelp:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Availability

Available in OS X v10.0 and later.

See Also

– `helpRequested:` (page 35)

Declared in

`NSResponder.h`

supplementalTargetForAction:sender:

This method is used in the process of finding a target for an action method.

```
- (id)supplementalTargetForAction:(SEL)action sender:(id)sender
```

Parameters

`action`

The requested action.

`sender`

The message sender.

Return Value

An object which responds to the action, or `nil`.

Discussion

If this `NSResponder` instance does not itself `respondsToSelector:`, then `supplementalTargetForAction:sender:` is called.

This method should return an object which responds to the action; if this responder does not have a supplemental object that does that, the implementation of this method should call super's `supplementalTargetForAction:sender:`.

© 2011 Apple Inc. All Rights Reserved.

NSResponder's implementation returns `nil`.

Availability
Available in OS X v10.7 and later.

Declared in
`NSResponder.h`

swapWithMark:

Swaps the mark and the selection or insertion point, so that what was marked is now the selection or insertion point, and what was the insertion point or selection is now the mark.

```
- (void)swapWithMark:(id)sender
```

Parameters
`sender`
Typically the object that invoked this method.

Discussion
`NSResponder` declares but doesn't implement this method.

Availability
Available in OS X v10.0 and later.

See Also
– `setMark:` (page 88)

Declared in
`NSResponder.h`

swipeWithEvent:

Informs the receiver that the user has begun a swipe gesture.

```
- (void)swipeWithEvent:(NSEvent *)event
```

Parameters
`event`
An event object representing the swipe gesture.

© 2011 Apple Inc. All Rights Reserved.

Discussion

The event will be sent to the view under the touch in the key window.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

tabletPoint:

Informs the receiver that a tablet-point event has occurred.

```
- (void)tabletPoint:(NSEvent *)theEvent
```

Parameters

`theEvent`

An object encapsulating information about the tablet-point event.

Discussion

Tablet events are represented by `NSEvent` objects of type `NSTabletPoint`. They describe the current state of a transducer (that is, a pointing device) that is in proximity to its tablet, reflecting changes such as location, pressure, tilt, and rotation. See the `NSEvent` reference for the methods that allow you to extract this and other information from `theEvent`. The default implementation of `NSResponder` passes the message to the next responder.

Availability

Available in OS X v10.4 or later.

See Also

– `tabletProximity:` (page 93)

Declared in

`NSResponder.h`

tabletProximity:

Informs the receiver that a tablet-proximity event has occurred.

```
- (void)tabletProximity:(NSEvent *)theEvent
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`theEvent`

An object encapsulating information about the tablet-point event.

Discussion

Tablet events are represented by `NSEvent` objects of type `NSTabletProximity`. Tablet devices generate proximity events when the transducer (pointing device) nears a tablet and when it moves away from a tablet. From an event object of this type you can extract information about the kind of device and its capabilities, as well as the relation of this tablet-proximity event to various tablet-point events; see the `NSEvent` reference for details. The default implementation passes the message to the next responder.

Availability

Available in OS X v10.4 or later.

See Also

`– tabletPoint:` (page 93)

Declared in

`NSResponder.h`

touchesBeganWithEvent:

Informs the receiver that new set of touches has been recognized.

```
– (void)touchesBeganWithEvent:(NSEvent *)event
```

Parameters

`event`

An event object representing the beginning of a touch.

Discussion

The event will be sent to the view under the touch in the key window. To get the set of touches that began for this view (or descendants of this view) send [event touchesMatchingPhase:NSTouchPhaseBegan inView:self].

This is not always the point of contact with the touch device. A touch that transitions from resting to active may be part of a `touchesBeganWithEvent:` set.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

touchesCancelledWithEvent:

Informs the receiver that tracking of touches has been cancelled for any reason..

```
- (void)touchesCancelledWithEvent:(NSEvent *)event
```

Parameters

`event`

An event object representing the cancellation of a touch event.

Discussion

The event will be sent to the view under the touch in the key window.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

touchesEndedWithEvent:

Returns that a set of touches have been removed.

```
- (void)touchesEndedWithEvent:(NSEvent *)event
```

Parameters

`event`

An event object representing the ending of a touch event.

Discussion

The event will be sent to the view under the touch in the key window. To get the set of touches that ended for this view (or descendants of this view) send [event touchesMatchingPhase:NSTouchPhaseEnded inView:self].

This is not always the point of removal with the touch device. A touch that transitions from active to resting may be part of an `touchesEndedWithEvent:` set.

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

touchesMovedWithEvent:

Informs the receiver that one or more touches has moved.

```
- (void)touchesMovedWithEvent:(NSEvent *)event
```

Parameters

`event`

An event object representing a touch movement.

Discussion

The event will be sent to the view under the touch in the key window. To get the set of touches that moved for this view (or descendants of this view) send [event touchesMatchingPhase:NSTouchPhaseMoved inView:self].

Availability

Available in OS X v10.6 and later.

Declared in

`NSResponder.h`

transpose:

Transposes the characters to either side of the insertion point and advances the insertion point past both of them. Does nothing to a selected range of text.

```
- (void)transpose:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

transposeWords:

Transposes the words to either side of the insertion point and advances the insertion point past both of them. Does nothing to a selected range of text.

```
- (void)transposeWords:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

`NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

Declared in

`NSResponder.h`

tryToPerform:with:

Attempts to perform the action indicated method with a specified argument.

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

Parameters

`anAction`

The selector identifying the action method.

`anObject`

The object to use as the sole argument of the action method.

Return Value

Returns `NO` if no responder is found that responds to `anAction`, `YES` otherwise.

Discussion

If the receiver responds to `anAction`, it invokes the method with `anObject` as the argument and returns `YES`. If the receiver doesn't respond, it sends this message to its next responder with the same selector and object.

Availability

Available in OS X v10.0 and later.

© 2011 Apple Inc. All Rights Reserved.

See Also

– doCommandBySelector: (page 32)
sendAction:to:from: (NSApplication)

Declared in

NSResponder.h

undoManager

Returns the undo manager for this responder.

```
– (NSUndoManager *)undoManager
```

Discussion

NSResponder's implementation simply passes this message to the next responder.

Availability

Available in OS X v10.0 and later.

Related Sample Code

DotViewUndo
Sketch
Sketch+Accessibility
TextEdit

Declared in

NSResponder.h

uppercaseWord:

Implemented by subclasses to make uppercase every letter in the word or words surrounding the insertion point or selection, expanding the selection if necessary.

```
– (void)uppercaseWord:(id)sender
```

Parameters

sender
: Typically the object that invoked this method.

Discussion

If either end of the selection partially covers a word, that entire word is made uppercase. NSResponder declares but doesn't implement this method.

© 2011 Apple Inc. All Rights Reserved.

Availability
Available in OS X v10.0 and later.

See Also
– lowercaseWord: (page 43)
– capitalizeWord: (page 24)
– changeCaseOfLetter: (page 25)

Declared in
NSResponder.h

validateProposedFirstResponder:forEvent:

Allow controls to determine when they should become first responder.

```
- (BOOL)validateProposedFirstResponder:(NSResponder *)responder forEvent:(NSEvent
*)event
```

Parameters
responder
: The first responder.

event
: The event to validate. May be nil if there is no applicable event.

Return Value
YES if the control should become first responder, otherwise NO.

Discussion
Some controls, such as NSTextField, should only become first responder when the enclosing NSTableView/NSBrowser indicates that the view can begin editing. It is up to the particular control that wants to be validated to call this method in its mouseDown: method (or perhaps at another time) to determine if it should attempt to become the first responder or not.

The NSTableView, NSOutlineView, and NSBrowser classes implement this to allow first responder status only if the responder is a view in a selected row. It also delays the first responder assignment if a doubleAction may be invoked.

The default implementation returns YES when there is no nextResponder (page 71) set, otherwise, it is forwarded up the responder chain.

Availability
Available in OS X v10.7 and later.

© 2011 Apple Inc. All Rights Reserved.

Declared in
`NSResponder.h`

validRequestorForSendType:returnType:

Overridden by subclasses to determine what services are available.

```
- (id)validRequestorForSendType:(NSString *)sendType returnType:(NSString *)returnType
```

Parameters

`sendType`
A string identifying the send type of pasteboard data. May be an empty string (see discussion).

`returnType`
A string identifying the return type of pasteboard data. May be an empty string (see discussion).

Return Value

If the receiver can place data of `sendType` on the pasteboard and receive data of `returnType`, it should return `self`; otherwise it should return either `[super validRequestorForSendType:returnType:]` or `[[self nextResponder] validRequestorForSendType:returnType:]`, which allows an object higher up in the responder chain to have an opportunity to handle the message.

Discussion

With each event, and for each service in the Services menu, the application object sends this message up the responder chain with the send and return type for the service being checked. This method is therefore invoked many times per event. The default implementation simply forwards this message to the next responder, ultimately returning `nil`.

Either `sendType` or `returnType`—but not both—may be empty. If `sendType` is empty, the service doesn't require input from the application requesting the service. If `returnType` is empty, the service doesn't return data.

Availability

Available in OS X v10.0 and later.

See Also

`registerServicesMenuSendTypes:returnTypes:` (`NSApplication`)
`writeSelectionToPasteboard:types:` (`NSServicesRequests` protocol)
`readSelectionFromPasteboard:` (`NSServicesRequests` protocol)

Declared in
`NSResponder.h`

© 2011 Apple Inc. All Rights Reserved.

wantsForwardedScrollEventsForAxis:

Returns whether to forward elastic scrolling gesture events up the responder.

```
- (BOOL)wantsForwardedScrollEventsForAxis:(NSEventGestureAxis)axis
```

Parameters

`axis`

The gesture axis. See `NSEventGestureAxis` for the possible values.

Return Value

Returns `YES` when forward gesture scroll events should be forwarded up the responder chain when the scrolling content is already at the edge of the scrolled direction at the beginning of the scroll gesture; `NO` otherwise.

Discussion

Some views process gesture scroll events to perform elastic scrolling. In some cases, you may want to track gesture scroll events like a swipe, see, `trackSwipeEventWithOptions:usingHandler:`.

Implement this method and return `YES` in your swipe controller and views that perform elastic scrolling will forward gesture scroll events up the responder chain on the following condition: the content to be scrolled is already at the edge of the scrolled direction at the beginning of the scroll gesture.

Otherwise, the view will perform elastic scrolling.

Availability

Available in OS X v10.7 and later.

Declared in

`NSResponder.h`

willPresentError:

Implemented by subclasses to return a custom version of the supplied error object that is more suitable for presentation in alert sheets and dialogs.

```
- (NSError *)willPresentError:(NSError *)anError
```

Parameters

`anError`

The error object to be customized.

Return Value

The customized error object; if you decide not to customize the error presentation, return by sending this message to `super` (that is, `return [super willPresentError:anError]`).

© 2011 Apple Inc. All Rights Reserved.

Discussion

When overriding this method, you can examine `anError` and, if its localized description or recovery information is unhelpfully generic, return an error object with more specific localized text. If you do this, always use the domain and error code of the `NSError` object to distinguish between errors whose presentation you want to customize and those you do not. Don't make decisions based on the localized description, recovery suggestion, or recovery options because parsing localized text is problematic.

The default implementation of this method simply returns `anError` unchanged.

Availability

Available in OS X v10.4 and later.

See Also

– `presentError:` (page 77)
– `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 78)

Declared in

`NSResponder.h`

yank:

Replaces the insertion point or selection with text from the kill buffer.

```
– (void)yank:(id)sender
```

Parameters

`sender`

Typically the object that invoked this method.

Discussion

If invoked sequentially, cycles through the kill buffer in reverse order. See "Standard Action Methods for Selecting and Editing" in *Cocoa Event Handling Guide* for more information on the kill buffer. `NSResponder` declares but doesn't implement this method.

Availability

Available in OS X v10.0 and later.

See Also

– `deleteToBeginningOfLine:` (page 28)
– `deleteToEndOfLine:` (page 29)
– `deleteToBeginningOfParagraph:` (page 29)
– `deleteToEndOfParagraph:` (page 30)

© 2011 Apple Inc. All Rights Reserved.

– deleteToMark: (page 30)

Declared in
NSResponder.h

© 2011 Apple Inc. All Rights Reserved.

Deprecated NSResponder Methods

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

Deprecated in OS X v10.8

interfaceStyle

Returns the receiver's interface style. (Deprecated in OS X v10.8.)

```
- (NSInterfaceStyle)interfaceStyle
```

Discussion

`interfaceStyle` is an abstract method in `NSResponder` and just returns `NSNoInterfaceStyle`. It is overridden in classes such as `NSWindow` and `NSView` to return the interface style, such as `NSMacintoshInterfaceStyle`. A responder's style (if other than `NSNoInterfaceStyle`) overrides all other settings, such as those established by the defaults system.

Availability

Available in OS X v10.0 and later.

Deprecated in OS X v10.8.

See Also

– `setInterfaceStyle:` (page 105)

Declared in

`NSInterfaceStyle.h`

performMnemonic:

Overridden by subclasses to handle a mnemonic. (Deprecated in OS X v10.8.)

```
- (BOOL)performMnemonic:(NSString *)aString
```

© 2011 Apple Inc. All Rights Reserved.

Parameters

`aString`

A string containing mnemonic character code or codes.

Discussion

If the character code or codes in `aString` match the receiver's mnemonic, the receiver should perform the mnemonic and return `YES`. The default implementation does nothing and returns `NO`. Mnemonics are not supported in OS X.

Availability

Available in OS X v10.0 and later.

Deprecated in OS X v10.8.

See Also

`performMnemonic:` (NSView)

Declared in

`NSResponder.h`

setInterfaceStyle:

Sets the receiver's style to the style specified by `interfaceStyle`, such as `NSMacintoshInterfaceStyle` or `NSWindows95InterfaceStyle`. (Deprecated in OS X v10.8.)

```
- (void)setInterfaceStyle:(NSInterfaceStyle)interfaceStyle
```

Parameters

`interfaceStyle`

An `enum` constant identifying an interface style.

Discussion

`setInterfaceStyle:` is an abstract method in `NSResponder`, but is overridden in classes such as `NSWindow` and `NSView` to actually set the interface style. You should almost never need to invoke or override this method, but if you do override it, your version should always invoke the implementation in `super`.

Availability

Available in OS X v10.0 and later.

Deprecated in OS X v10.8.

See Also

– `interfaceStyle` (page 104)

© 2011 Apple Inc. All Rights Reserved.

Declared in

`NSInterfaceStyle.h`

© 2011 Apple Inc. All Rights Reserved.

Document Revision History

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

Date	Notes
2011-06-06	Updated for OS X v10.7.
2010-02-18	Added note regarding alternate behavior of rightMouseDown: in NSView.
2009-10-05	Corrected swipeWithEvent: description.
2009-06-01	Updated to OS X v10.6. New support for touches, bi-directional text, and additional key bindings.
2007-03-05	Updated for OS X version 10.5.
2006-12-05	Corrected the wording of acceptsFirstResponder, insertText:, and cancelOperation:.
2006-05-23	First publication of this content as a separate document.

© 2011 Apple Inc. All Rights Reserved.

Apple Inc.
© 2011 Apple Inc.
All rights reserved.

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

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

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

Apple, the Apple logo, Cocoa, Finder, Mac, Macintosh, and OS X are trademarks of Apple Inc., registered in the U.S. and other countries.

OpenCL is a trademark of Apple Inc.

OpenGL is a registered trademark of Silicon Graphics, Inc.

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

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

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

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