NSResponder Class Reference

Cocoa > Events & Other Input

2007-03-05

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

Simultaneously published in the United States and Canada.

Contents

NSResponder Class Reference 7

Overview 7
Adopted Protocols 8
Tasks 8
- Changing the First Responder 8
- Managing the Next Responder 8
- Responding to Mouse Events 8
- Responding to Key Events 9
- Responding to Other Kinds of Events 9
- Responding to Action Messages 10
- Presenting and Customizing Error Information 14
- Dispatching Messages 14
- Managing a Responder's Menu 14
- Updating the Services Menu 15
- Getting the Undo Manager 15
- Testing Events 15
- Terminating the Responder Chain 15
- Setting the Interface Style 15

Instance Methods 15
- acceptsFirstResponder 15
- becomeFirstResponder 16
- cancelOperation: 16
- capitalizeWord: 17
- centerSelectionInVisibleArea: 17
- changeCaseOfLetter: 18
- complete: 18
- cursorUpdate: 19
- deleteBackward: 19
- deleteBackwardByDecomposingPreviousCharacter: 20
- deleteForward: 20
- deleteToBeginningOfLine: 20
- deleteToBeginningOfParagraph: 21
- deleteToEndOfLine: 21
- deleteToEndOfParagraph: 22
- deleteToMark: 22
- deleteWordBackward: 23
- deleteWordForward: 23
- doCommandBySelector: 23
- flagsChanged: 24
- flushBufferedKeyEvents 24
- helpRequested: 24

indent: 25
insertBacktab: 25
insertContainerBreak: 26
insertLineBreak: 26
insertNewline: 26
insertNewlineIgnoringFieldEditor: 27
insertParagraphSeparator: 27
insertTab: 27
insertTabIgnoringFieldEditor: 28
insertText: 28
interfaceStyle 29
interpretKeyEvents: 29
keyDown: 29
keyUp: 30
lowercaseWord: 30
menu 31
mouseDown: 31
mouseDragged: 32
mouseEntered: 32
mouseExited: 32
mouseMoved: 33
mouseUp: 33
moveBackward: 33
moveBackwardAndModifySelection: 34
moveDown: 34
moveDownAndModifySelection: 35
moveForward: 35
moveForwardAndModifySelection: 36
moveLeft: 36
moveLeftAndModifySelection: 36
moveRight: 37
moveRightAndModifySelection: 37
moveToBeginningOfDocument: 38
moveToBeginningOfLine: 38
moveToBeginningOfParagraph: 39
moveToEndOfDocument: 39
moveToEndOfLine: 39
moveToEndOfParagraph: 40
moveUp: 40
moveUpAndModifySelection: 41
moveWordBackward: 41
moveWordBackwardAndModifySelection: 41
moveWordForward: 42
moveWordForwardAndModifySelection: 42
moveWordLeft: 43
moveWordLeftAndModifySelection: 43

moveWordRight: 44
moveWordRightAndModifySelection: 44
nextResponder 45
noResponderFor: 45
otherMouseDown: 46
otherMouseDragged: 46
otherMouseUp: 46
pageDown: 47
pageUp: 47
performKeyEquivalent: 48
performMnemonic: 48
presentError: 49
presentError:modalForWindow:delegate:didPresentSelector:contextInfo: 49
resignFirstResponder 50
rightMouseDown: 51
rightMouseDragged: 51
rightMouseUp: 52
scrollLineDown: 52
scrollLineUp: 53
scrollPageDown: 53
scrollPageUp: 54
scrollWheel: 54
selectAll: 54
selectLine: 55
selectParagraph: 55
selectToMark: 55
selectWord: 56
setInterfaceStyle: 56
setMark: 57
setMenu: 57
setNextResponder: 58
shouldBeTreatedAsInkEvent: 58
showContextHelp: 59
swapWithMark: 59
tabletPoint: 59
tabletProximity: 60
transpose: 60
transposeWords: 61
tryToPerform:with: 61
undoManager 62
uppercaseWord: 62
validRequestorForSendType:returnType: 62
willPresentError: 63
yank: 64

Document Revision History 65

Index 67

NSResponder Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Cocoa Event-Handling Guide
Declared in	NSInterfaceStyle.h NSResponder.h
Related sample code	JSPong LayerBackedOpenGLView NSOpenGL Fullscreen

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 Mac OS X v10.4, `NSResponder` plays an important role in the presentation of error information. The default implementations of the `presentError:` (page 49) and `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 49) methods send `willPresentError:` (page 63) to `self`, thereby giving subclasses the opportunity to customize the localized 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.)

Adopted Protocols

NSCoding
- `encodeWithCoder:`
- `initWithCoder:`

Tasks

Changing the First Responder

- `acceptsFirstResponder` (page 15)
  Overridden by subclasses to return `YES` if the receiver accepts first responder status.
- `becomeFirstResponder` (page 16)
  Notifies the receiver that it's about to become first responder in its `NSWindow`.
- `resignFirstResponder` (page 50)
  Notifies the receiver that it's been asked to relinquish its status as first responder in its window.

Managing the Next Responder

- `setNextResponder:` (page 58)
  Sets the receiver's next responder.
- `nextResponder` (page 45)
  Returns the receiver's next responder, or `nil` if it has none.

Responding to Mouse Events

- `mouseDown:` (page 31)
  Informs the receiver that the user has pressed the left mouse button.
- `mouseDragged:` (page 32)
  Informs the receiver that the user has moved the mouse with the left button pressed.
- `mouseUp:` (page 33)
  Informs the receiver that the user has released the left mouse button.
- `mouseMoved:` (page 33)
  Informs the receiver that the mouse has moved specified.
- `mouseEntered:` (page 32)
  Informs the receiver that the cursor has entered a tracking rectangle.

- `mouseExited:` (page 32)
  Informs the receiver that the cursor has exited a tracking rectangle.
- `rightMouseDown:` (page 51)
  Informs the receiver that the user has pressed the right mouse button.
- `rightMouseDragged:` (page 51)
  Informs the receiver that the user has moved the mouse with the right button pressed .
- `rightMouseUp:` (page 52)
  Informs the receiver that the user has released the right mouse button.
- `otherMouseDown:` (page 46)
  Informs the receiver that the user has pressed a mouse button other than the left or right one.
- `otherMouseDragged:` (page 46)
  Informs the receiver that the user has moved the mouse with a button other than the left or right button pressed.
- `otherMouseUp:` (page 46)
  Informs the receiver that the user has released a mouse button other than the left or right button.

Responding to Key Events

- `keyDown:` (page 29)
  Informs the receiver that the user has pressed a key.
- `keyUp:` (page 30)
  Informs the receiver that the user has released a key.
- `interpretKeyEvents:` (page 29)
  Invoked by subclasses from their `keyDown:` (page 29) method to handle a series of key events.
- `performKeyEquivalent:` (page 48)
  Overridden by subclasses to handle a key equivalent.
- `performMnemonic:` (page 48)
  Overridden by subclasses to handle a mnemonic.
- `flushBufferedKeyEvents` (page 24)
  Overridden by subclasses to clear any unprocessed key events.

Responding to Other Kinds of Events

- `cursorUpdate:` (page 19)
  Informs the receiver that the mouse cursor has moved into a cursor rectangle.
- `flagsChanged:` (page 24)
  Informs the receiver that the user has pressed or released a modifier key (Shift, Control, and so on).
- `tabletPoint:` (page 59)
  Informs the receiver that a tablet-point event has occurred.
- `tabletProximity:` (page 60)
  Informs the receiver that a tablet-proximity event has occurred.
- `helpRequested:` (page 24)
  Displays context-sensitive help for the receiver if such exists; otherwise passes this message to the next responder.

- `scrollWheel:` (page 54)
  Informs the receiver that the mouse's scroll wheel has moved.

Responding to Action Messages

- `cancelOperation:` (page 16)
  Implemented by subclasses to cancel the current operation.
- `capitalizeWord:` (page 17)
  Implemented by subclasses to capitalize the word or words surrounding the insertion point or selection, expanding the selection if necessary.
- `centerSelectionInVisibleArea:` (page 17)
  Implemented by subclasses to scroll the selection, whatever it is, inside its visible area.
- `changeCaseOfLetter:` (page 18)
  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 18)
  Implemented by subclasses to complete an operation in progress or a partially constructed element.
- `deleteBackward:` (page 19)
  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 20)
  Implemented by subclasses to delete the selection, if there is one, or a single character backward from the insertion point.
- `deleteForward:` (page 20)
  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 20)
  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 21)
  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 21)
  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 22)
  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 22)
  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 23)
  Implemented by subclasses to delete the selection, if there is one, or a single word backward from the insertion point.

- `deleteWordForward:` (page 23)
  Implemented by subclasses to delete the selection, if there is one, or a single word forward from the insertion point.
- `indent:` (page 25)
  Implemented by subclasses to indent the selection or the insertion point if there is no selection.
- `insertBacktab:` (page 25)
  Implemented by subclasses to handle a backward tab."
- `insertContainerBreak:` (page 26)
  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 26)
  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 26)
  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 27)
  Implemented by subclasses to insert a line-break character at the insertion point or selection, deleting the selection if there is one.
- `insertParagraphSeparator:` (page 27)
  Implemented by subclasses to insert a paragraph separator at the insertion point or selection, deleting the selection if there is one.
- `insertTab:` (page 27)
  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 28)
  Implemented by subclasses to insert a tab character at the insertion point or selection, deleting the selection if there is one.
- `insertText:` (page 28)
  Overridden by subclasses to insert the supplied string at the insertion point or selection, deleting the selection if there is one.
- `lowercaseWord:` (page 30)
  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 33)
  Implemented by subclasses to move the selection or insertion point one element or character backward.
- `moveBackwardAndModifySelection:` (page 34)
  Implemented by subclasses to expand or reduce either end of the selection backward by one element or character.
- `moveDown:` (page 34)
  Implemented by subclasses to move the selection or insertion point one element or character down.
- `moveDownAndModifySelection:` (page 35)
  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 35)
  Implemented by subclasses to move the selection or insertion point one element or character forward.

- `moveForwardAndModifySelection:` (page 36)
  Implemented by subclasses to expand or reduce either end of the selection forward by one element or character.
- `moveLeft:` (page 36)
  Implemented by subclasses to move the selection or insertion point one element or character to the left.
- `moveLeftAndModifySelection:` (page 36)
  Implemented by subclasses to expand or reduce either end of the selection to the left (display order) by one element or character.
- `moveRight:` (page 37)
  Implemented by subclasses to move the selection or insertion point one element or character to the right.
- `moveRightAndModifySelection:` (page 37)
  Implemented by subclasses to expand or reduce either end of the selection to the right (display order) by one element or character.
- `moveToBeginningOfDocument:` (page 38)
  Implemented by subclasses to move the selection to the first element of the document or the insertion point to the beginning.
- `moveToBeginningOfLine:` (page 38)
  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.
- `moveToBeginningOfParagraph:` (page 39)
  Implemented by subclasses to move the insertion point to the beginning of the selected paragraph.
- `moveToEndOfDocument:` (page 39)
  Implemented by subclasses to move the selection to the last element of the document or the insertion point to the end.
- `moveToEndOfLine:` (page 39)
  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 40)
  Implemented by subclasses to move the insertion point to the end of the selected paragraph.
- `moveUp:` (page 40)
  Implemented by subclasses to move the selection or insertion point one element or character up.
- `moveUpAndModifySelection:` (page 41)
  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 41)
  Implemented by subclasses to move the selection or insertion point one word backward.
- `moveWordBackwardAndModifySelection:` (page 41)
  Implemented by subclasses to expand or reduce either end of the selection backward by one whole word.
- `moveWordForward:` (page 42)
  Implemented by subclasses to move the selection or insertion point one word forward, in logical order.

- `moveWordForwardAndModifySelection:` (page 42)
  Implemented by subclasses to expand or reduce either end of the selection forward by one whole word.
- `moveWordLeft:` (page 43)
  Implemented by subclasses to move the selection or insertion point one word to the left, in display order.
- `moveWordRight:` (page 44)
  Implemented by subclasses to move the selection or insertion point one word right.
- `moveWordRightAndModifySelection:` (page 44)
  Implemented by subclasses to expand or reduce either end of the selection to the right by one whole word.
- `moveWordLeftAndModifySelection:` (page 43)
  Implemented by subclasses to expand or reduce either end of the selection left by one whole word in display order.
- `pageDown:` (page 47)
  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.
- `pageUp:` (page 47)
  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.
- `scrollLineDown:` (page 52)
  Implemented by subclasses to scroll the receiver one line down in its scroll view, without changing the selection.
- `scrollLineUp:` (page 53)
  Implemented by subclasses to scroll the receiver one line up in its scroll view, without changing the selection.
- `scrollPageDown:` (page 53)
  Implemented by subclasses to scroll the receiver one page down in its scroll view, without changing the selection.
- `scrollPageUp:` (page 54)
  Implemented by subclasses to scroll the receiver one page up in its scroll view, without changing the selection.
- `selectAll:` (page 54)
  Implemented by subclasses to select all selectable elements.
- `selectLine:` (page 55)
  Implemented by subclasses to select all elements in the line or lines containing the selection or insertion point.
- `selectParagraph:` (page 55)
  Implemented by subclasses to select all paragraphs containing the selection or insertion point.
- `selectToMark:` (page 55)
  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 56)
  Implemented by subclasses to extend the selection to the nearest word boundaries outside it (up to, but not including, word delimiters).

- `setMark:` (page 57)
  Implemented by subclasses to set a mark at the insertion point or selection, which is used by `deleteToMark:` (page 22) and `selectToMark:` (page 55).
- `showContextHelp:` (page 59)
  Implemented by subclasses to invoke the help system, displaying information relevant to the receiver and its current state.
- `swapWithMark:` (page 59)
  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 60)
  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 61)
  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 62)
  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 64)
  Replaces the insertion point or selection with text from the kill buffer.

Presenting and Customizing Error Information

- `presentError:` (page 49)
  Presents an error alert to the user as an application-modal dialog.
- `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 49)
  Presents an error alert to the user as a document-modal sheet attached to document window.
- `willPresentError:` (page 63)
  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 23)
  Attempts to perform the indicated command.
- `tryToPerform:with:` (page 61)
  Attempts to perform the action indicated method with a specified argument.

Managing a Responder's Menu

- `setMenu:` (page 57)
  Sets the receiver's menu.
- `menu` (page 31)
  Returns the receiver's menu.

Updating the Services Menu

- `validRequestorForSendType:returnType:` (page 62)
  Overridden by subclasses to determine what services are available.

Getting the Undo Manager

- `undoManager` (page 62)
  Returns the undo manager for this responder.

Testing Events

- `shouldBeTreatedAsInkEvent:` (page 58)
  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 45)
  Handles the case where an event or action message falls off the end of the responder chain.

Setting the Interface Style

- `setInterfaceStyle:` (page 56)
  Sets the receiver's style to the style specified by *`interfaceStyle`*, such as `NSMacintoshInterfaceStyle` or `NSWindows95InterfaceStyle`.
- `interfaceStyle` (page 29)
  Returns the receiver's interface style.

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 Mac OS X v10.0 and later.

See Also
- `becomeFirstResponder` (page 16)
- `resignFirstResponder` (page 50)
- `needsPanelToBecomeKey` (NSView)

Related Sample Code
Dicey
NSOpenGL Fullscreen
NURBSSurfaceVertexProg
Sketch-112
SurfaceVertexProgram

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.

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

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

See Also
- `resignFirstResponder` (page 50)
- `acceptsFirstResponder` (page 15)

Related Sample Code
Dicey
GLChildWindowDemo
NURBSSurfaceVertexProg
Sketch-112
SurfaceVertexProgram

Declared In
`NSResponder.h`

cancelOperation:

Implemented by subclasses to cancel the current operation.

```
- (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 Mac 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 Mac OS X v10.0 and later.

See Also

- `lowercaseWord:` (page 30)
- `uppercaseWord:` (page 62)
- `changeCaseOfLetter:` (page 18)

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 Mac OS X v10.0 and later.

See Also

- `- scrollLineDown:` (page 52)
- `- scrollLineUp:` (page 53)
- `- scrollPageDown:` (page 53)
- `- scrollPageUp:` (page 54)

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.

Discussion

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

Availability

Available in Mac OS X v10.0 and later.

See Also

- `- lowercaseWord:` (page 30)
- `- uppercaseWord:` (page 62)
- `- capitalizeWord:` (page 17)

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 Mac 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
```

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 Mac 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 Mac 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
```

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 Mac 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 Mac 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
```

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 Mac OS X v10.0 and later.

See Also

`- yank:` (page 64)

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 Mac OS X v10.0 and later.

See Also

`- yank:` (page 64)

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).

```
- (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 Mac 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 Mac OS X v10.0 and later.

See Also
- yank: (page 64)

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.

```
- (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 Mac OS X v10.0 and later.

See Also
- setMark: (page 57)
- selectToMark: (page 55)
- yank: (page 64)

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 Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

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 Mac 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 Mac OS X v10.0 and later.

See Also
- `tryToPerform:with:` (page 61)
- `sendAction:to:from:` (NSApplication)

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 Mac 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 Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

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 Mac OS X v10.0 and later.

See Also
`- showContextHelp:` (page 59)

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 Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

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 Mac 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 Mac OS X v10.4 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 Mac OS X v10.4 and later.

Declared In

`NSResponder.h`

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 Mac 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 26), 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 Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

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 Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

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 Mac 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 27), 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 Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

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 29) message. The `NSResponder` implementation simply passes this message to the next responder, or beeps if there is no next responder.

Availability

Available in Mac OS X v10.0 and later.

Declared In
NSResponder.h

interfaceStyle

Returns the receiver's interface style.

```
- (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 Mac OS X v10.0 and later.

See Also
- setInterfaceStyle: (page 56)

Declared In
NSInterfaceStyle.h

interpretKeyEvents:

Invoked by subclasses from their keyDown: (page 29) 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 28) and doCommandBySelector: (page 23) 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.

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

Declared In
NSResponder.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 29). The default implementation simply passes this message to the next responder.

Availability

Available in Mac OS X v10.0 and later.

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 Mac 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.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `uppercaseWord:` (page 62)
- `capitalizeWord:` (page 17)
- `changeCaseOfLetter:` (page 18)

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 Mac OS X v10.0 and later.

See Also
- `setMenu:` (page 57)
- `menuForEvent:` (NSView)
- `defaultMenu` (NSView)

Related Sample Code
MenuItemView
UIElementInspector

Declared In
`NSResponder.h`

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 Mac OS X v10.0 and later.

Related Sample Code
Sketch-112

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 Mac OS X v10.0 and later.

Declared In
NSResponder.h

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 Mac 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 Mac OS X v10.0 and later.

Declared In
NSResponder.h

mouseMoved:

Informs the receiver that the mouse has moved specified.

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

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 Mac 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 Mac OS X v10.0 and later.

Declared In
NSResponder.h

moveBackward:

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

```
- (void)moveBackward:(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 beginning of the former selection. `NSResponder` declares but doesn't implement this method.

Availability

Available in Mac 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 36) 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 36) is bound to the left arrow key.

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

Availability

Available in Mac 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.

```
- (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 Mac 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 41) 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 Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

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 Mac 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 34) 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 37) is bound to the right arrow key.

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

Availability

Available in Mac OS X v10.0 and later.

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 Mac 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:` (page 37) 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.

The essential difference between this method and the corresponding `moveBackwardAndModifySelection:` (page 34) 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 Mac OS X v10.3 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 Mac 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
```

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 36) 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 36) 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 Mac 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 Mac OS X v10.0 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 Mac OS X v10.0 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 Mac OS X v10.0 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 Mac OS X v10.0 and later.

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 Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

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 Mac OS X v10.0 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 Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

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 35) 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 Mac 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 Mac OS X v10.0 and later.

See Also

`- moveWordLeft:` (page 43)

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 42) 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 Mac OS X v10.0 and later.

See Also

`- moveWordLeftAndModifySelection:` (page 43)

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.

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 Mac OS X v10.0 and later.

See Also

`- moveWordRight:` (page 44)

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 41) 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 Mac OS X v10.0 and later.

See Also

`- moveWordRightAndModifySelection:` (page 44)

Declared In

`NSResponder.h`

moveWordLeft:

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

```
- (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 41) 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 Mac 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 44) 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 41) 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 Mac 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 42) 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 Mac 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.

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 41) 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 42) 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 Mac 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 Mac OS X v10.0 and later.

See Also

- `setNextResponder:` (page 58)
- `noResponderFor:` (page 45)

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.

Discussion

The default implementation beeps if *eventSelector* is `keyDown:` (page 29).

Availability

Available in Mac 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 Mac 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.

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

See Also
- scrollPageDown: (page 53)
- scrollPageUp: (page 54)

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.

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

See Also
- scrollPageDown: (page 53)
- scrollPageUp: (page 54)

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.

> **Note:** performKeyEquivalent: (page 48) takes an NSEvent object as its argument, while performMnemonic: (page 48) 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 Mac OS X v10.0 and later.

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

Declared In
NSResponder.h

performMnemonic:

Overridden by subclasses to handle a mnemonic.

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

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 Mac OS X.

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

See Also
- performMnemonic: (NSView)

Declared In
NSResponder.h

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 63) 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 63) instead.

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

See Also
- presentError:modalForWindow:delegate:didPresentSelector:contextInfo: (page 49)

Related Sample Code
Core Data HTML Store
CoreRecipes

Declared In
NSResponder.h

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 63) 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 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 63) instead.

Availability

Available in Mac OS X v10.4 and later.

See Also

- presentError: (page 49)

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 Mac OS X v10.0 and later.

See Also
- `becomeFirstResponder` (page 16)
- `acceptsFirstResponder` (page 15)

Related Sample Code
Clock Control
Dicey
GLChildWindowDemo
NURBSSurfaceVertexProg
SurfaceVertexProgram

Declared In
`NSResponder.h`

rightMouseDown:

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

```
- (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.

Availability
Available in Mac 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 Mac 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
```

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 Mac OS X v10.0 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.

Discussion

NSResponder declares but doesn't implement this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- scrollLineUp: (page 53)
- 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 Mac OS X v10.0 and later.

See Also
- scrollLineDown: (page 52)
- 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
```

Parameters

sender
Typically the object that invoked this method.

Discussion
NSResponder declares but doesn't implement this method.

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

See Also
- pageDown: (page 47)
- pageUp: (page 47)
- 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 Mac OS X v10.0 and later.

See Also

- `pageDown:` (page 47)
- `pageUp:` (page 47)
- `pageScroll` (NSScrollView)

Declared In

`NSResponder.h`

scrollWheel:

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

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

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 Mac 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 Mac 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.

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

Availability
Available in Mac 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 Mac 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.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setMark:` (page 57)
- `deleteToMark:` (page 22)

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 Mac OS X v10.0 and later.

Declared In

`NSResponder.h`

setInterfaceStyle:

Sets the receiver's style to the style specified by *interfaceStyle*, such as `NSMacintoshInterfaceStyle` or `NSWindows95InterfaceStyle`.

```
- (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 Mac OS X v10.0 and later.

See Also
- `interfaceStyle` (page 29)

Declared In
`NSInterfaceStyle.h`

setMark:

Implemented by subclasses to set a mark at the insertion point or selection, which is used by `deleteToMark:` (page 22) and `selectToMark:` (page 55).

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

Parameters
sender
Typically the object that invoked this method.

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

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

See Also
- `swapWithMark:` (page 59)

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 Mac OS X v10.0 and later.

See Also
- `menu` (page 31)

Related Sample Code
VertexPerformanceTest

Declared In
NSResponder.h

setNextResponder:

Sets the receiver's next responder.

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

Parameters

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

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

See Also
- nextResponder (page 45)

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 Mac OS X v10.2 and later.

Declared In
NSResponder.h

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 Mac OS X v10.0 and later.

See Also
- helpRequested: (page 24)

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 Mac OS X v10.0 and later.

See Also
- setMark: (page 57)

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 Mac OS X v10.4 or later.

See Also
`- tabletProximity:` (page 60)

Declared In
`NSResponder.h`

tabletProximity:

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

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

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 Mac OS X v10.4 or later.

See Also
`- tabletPoint:` (page 59)

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 Mac OS X v10.0 and later.

Declared In
`NSResponder.h`

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 Mac 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 Mac OS X v10.0 and later.

See Also
`- doCommandBySelector:` (page 23)

- 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 Mac OS X v10.0 and later.

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.

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

See Also
- lowercaseWord: (page 30)
- capitalizeWord: (page 17)
- changeCaseOfLetter: (page 18)

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 Mac OS X v10.0 and later.

See Also

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

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]`).

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 Mac OS X v10.4 and later.

See Also
- `presentError:` (page 49)
- `presentError:modalForWindow:delegate:didPresentSelector:contextInfo:` (page 49)

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" for more information on the kill buffer. `NSResponder` declares but doesn't implement this method.

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

See Also
- `deleteToBeginningOfLine:` (page 20)
- `deleteToEndOfLine:` (page 21)
- `deleteToBeginningOfParagraph:` (page 21)
- `deleteToEndOfParagraph:` (page 22)
- `deleteToMark:` (page 22)

Declared In
`NSResponder.h`

Document Revision History

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

Date	Notes
2007-03-05	Updated for Mac 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.

Document Revision History

Index

A

acceptsFirstResponder instance method 15

B

becomeFirstResponder instance method 16

C

cancelOperation: instance method 16
capitalizeWord: instance method 17
centerSelectionInVisibleArea: instance method 17
changeCaseOfLetter: instance method 18
complete: instance method 18
cursorUpdate: instance method 19

D

deleteBackwardByDecomposingPreviousCharacter: instance method 20
deleteBackward: instance method 19
deleteForward: instance method 20
deleteToBeginningOfLine: instance method 20
deleteToBeginningOfParagraph: instance method 21
deleteToEndOfLine: instance method 21
deleteToEndOfParagraph: instance method 22
deleteToMark: instance method 22
deleteWordBackward: instance method 23
deleteWordForward: instance method 23
doCommandBySelector: instance method 23

F

flagsChanged: instance method 24
flushBufferedKeyEvents instance method 24

H

helpRequested: instance method 24

I

indent: instance method 25
insertBacktab: instance method 25
insertContainerBreak: instance method 26
insertLineBreak: instance method 26
insertNewline: instance method 26
insertNewlineIgnoringFieldEditor: instance method 27
insertParagraphSeparator: instance method 27
insertTab: instance method 27
insertTabIgnoringFieldEditor: instance method 28
insertText: instance method 28
interfaceStyle instance method 29
interpretKeyEvents: instance method 29

K

keyDown: instance method 29
keyUp: instance method 30

L

lowercaseWord: instance method 30

M

menu instance method 31
mouseDown: instance method 31
mouseDragged: instance method 32
mouseEntered: instance method 32
mouseExited: instance method 32
mouseMoved: instance method 33
mouseUp: instance method 33
moveBackwardAndModifySelection: instance method 34
moveBackward: instance method 33
moveDownAndModifySelection: instance method 35
moveDown: instance method 34
moveForwardAndModifySelection: instance method 36
moveForward: instance method 35
moveLeftAndModifySelection: instance method 36
moveLeft: instance method 36
moveRightAndModifySelection: instance method 37
moveRight: instance method 37
moveToBeginningOfDocument: instance method 38
moveToBeginningOfLine: instance method 38
moveToBeginningOfParagraph: instance method 39
moveToEndOfDocument: instance method 39
moveToEndOfLine: instance method 39
moveToEndOfParagraph: instance method 40
moveUpAndModifySelection: instance method 41
moveUp: instance method 40
moveWordBackwardAndModifySelection: instance method 41
moveWordBackward: instance method 41
moveWordForwardAndModifySelection: instance method 42
moveWordForward: instance method 42
moveWordLeftAndModifySelection: instance method 43
moveWordLeft: instance method 43
moveWordRightAndModifySelection: instance method 44
moveWordRight: instance method 44

N

nextResponder instance method 45
noResponderFor: instance method 45

O

otherMouseDown: instance method 46
otherMouseDragged: instance method 46
otherMouseUp: instance method 46

P

pageDown: instance method 47
pageUp: instance method 47
performKeyEquivalent: instance method 48
performMnemonic: instance method 48
presentError: instance method 49
presentError:modalForWindow:delegate:didPresentSelector:contextInfo: instance method 49

R

resignFirstResponder instance method 50
rightMouseDown: instance method 51
rightMouseDragged: instance method 51
rightMouseUp: instance method 52

S

scrollLineDown: instance method 52
scrollLineUp: instance method 53
scrollPageDown: instance method 53
scrollPageUp: instance method 54
scrollWheel: instance method 54
selectAll: instance method 54
selectLine: instance method 55
selectParagraph: instance method 55
selectToMark: instance method 55
selectWord: instance method 56
setInterfaceStyle: instance method 56
setMark: instance method 57
setMenu: instance method 57
setNextResponder: instance method 58
shouldBeTreatedAsInkEvent: instance method 58
showContextHelp: instance method 59
swapWithMark: instance method 59

T

tabletPoint: instance method 59
tabletProximity: instance method 60
transpose: instance method 60
transposeWords: instance method 61

tryToPerform:with: instance method 61

U

undoManager instance method 62
uppercaseWord: instance method 62

V

validRequestorForSendType:returnType: instance method 62

W

willPresentError: instance method 63

Y

yank: instance method 64