NStext Class Reference

Cocoa > Text & Fonts

2008-05-06

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

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

Simultaneously published in the United States and Canada.

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

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

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

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

Contents

NSText Class Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSChangeSpelling NSIgnoreMisspelledWords NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Text System Overview
Declared in	NSText.h
Related sample code	EnhancedAudioBurn EnhancedDataBurn ImageBackground QTKitMovieShuffler QTSSInspector

Class at a Glance

`NSText` declares the most general programmatic interface for objects that manage text. You usually use instances of its subclass, `NSTextView`.

Principal Attributes

- Draws text for user interface objects
- Uses a delegate
- Provides text editing capabilities
- Controls text attributes such as typesize, font, and color

Commonly Used Methods

`readRTFDFromFile:` (page 25)
: Reads an `.rtf` or `.rtfd` file.

`writeRTFDToFile:atomically:` (page 42)
: Writes the receiver's text to a file.

`string` (page 39)
: Returns the receiver's text without attributes.

`RTFFromRange:` (page 28)
: Returns the receiver's text with attributes.

`RTFDFromRange:` (page 27)
: Returns the receiver's text with attributes and attachments.

Overview

`NSText` declares the most general programmatic interface for objects that manage text. You usually use instances of its subclass, `NSTextView`.

`NSTextView` extends the interface declared by `NSText` and provides much more sophisticated functionality than that declared in `NSText`.

`NSText` initialization creates an instance of a concrete subclass, such as `NSTextView`. Instances of any of these classes are generically called text objects.

Text objects are used by the Application Kit wherever text appears in interface objects: A text object draws the title of a window, the commands in a menu, the title of a button, and the items in a browser. Your application can also create text objects for its own purposes.

Adopted Protocols

NSChangeSpelling
- `changeSpelling:`

NSIgnoreMisspelledWords
- `ignoreSpelling:`

Tasks

Getting the Characters

`– string` (page 39)
: Returns the characters of the receiver's text.

Setting Graphics Attributes

- `setBackgroundColor:` (page 30)
  Sets the receiver's background color to a given color.
- `backgroundColor` (page 16)
  Returns the receiver's background color.
- `setDrawsBackground:` (page 31)
  Controls whether the receiver draws its background.
- `drawsBackground` (page 19)
  Returns a Boolean value that indicates whether the receiver draws its background.

Setting Behavioral Attributes

- `setEditable:` (page 31)
  Controls whether the receiver allows the user to edit its text.
- `isEditable` (page 21)
  Returns a Boolean value that indicates whether the receiver allows the user to edit text, `NO` if it doesn't.
- `setSelectable:` (page 35)
  Controls whether the receiver allows the user to select its text.
- `isSelectable` (page 23)
  Returns a Boolean value that indicates whether the receiver allows the user to select text, `NO` if it doesn't.
- `setFieldEditor:` (page 32)
  Controls whether the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder.
- `isFieldEditor` (page 21)
  Returns a Boolean value that indicates whether the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder.
- `setRichText:` (page 35)
  Controls whether the receiver allows the user to apply attributes to specific ranges of the text.
- `isRichText` (page 22)
  Returns a Boolean value that indicates whether the receiver allows the user to apply attributes to specific ranges of the text.
- `setImportsGraphics:` (page 33)
  Controls whether the receiver allows the user to import files by dragging.
- `importsGraphics` (page 20)
  Returns a Boolean value that indicates whether the receiver allows the user to import files by dragging.

Using the Font Panel and Menu

- `setUsesFontPanel:` (page 37)
  Controls whether the receiver uses the Font panel and Font menu.
- `usesFontPanel` (page 42)
  Returns a Boolean value that indicates whether the receiver uses the Font panel.

Using the Ruler

`- toggleRuler:` (page 41)
: This action method shows or hides the ruler, if the receiver is enclosed in a scroll view.

`- isRulerVisible` (page 22)
: Returns a Boolean value that indicates whether the receiver's enclosing scroll view shows its ruler.

Changing the Selection

`- setSelectedRange:` (page 36)
: Selects the receiver's characters within *`aRange`*.

`- selectedRange` (page 29)
: Returns the range of selected characters.

Replacing Text

`- replaceCharactersInRange:withRTF:` (page 25)
: Replaces the characters in the given range with RTF text interpreted from the given RTF data.

`- replaceCharactersInRange:withRTFD:` (page 26)
: Replaces the characters in the given range with RTFD text interpreted from the given RTFD data.

`- replaceCharactersInRange:withString:` (page 27)
: Replaces the characters in the given range with those in the given string.

`- setString:` (page 36)
: Replaces the receiver's entire text with *`aString`*, applying the formatting attributes of the old first character to its new contents.

Action Methods for Editing

`- selectAll:` (page 28)
: This action method selects all of the receiver's text.

`- copy:` (page 17)
: This action method copies the selected text onto the general pasteboard, in as many formats as the receiver supports.

`- cut:` (page 18)
: This action method deletes the selected text and places it onto the general pasteboard, in as many formats as the receiver supports.

`- paste:` (page 24)
: This action method pastes text from the general pasteboard at the insertion point or over the selection.

`- copyFont:` (page 18)
: This action method copies the font information for the first character of the selection (or for the insertion point) onto the font pasteboard, as `NSFontPboardType`.

`- pasteFont:` (page 24)
: This action method pastes font information from the font pasteboard onto the selected text or insertion point of a rich text object, or over all text of a plain text object.

- `copyRuler:` (page 18)

  This action method copies the paragraph style information for first selected paragraph onto the ruler pasteboard, as `NSRulerPboardType`, and expands the selection to paragraph boundaries.

- `pasteRuler:` (page 25)

  This action method pastes paragraph style information from the ruler pasteboard onto the selected paragraphs of a rich text object.

- `delete:` (page 19)

  This action method deletes the selected text.

Changing the Font

- `changeFont:` (page 16)

  This action method changes the font of the selection for a rich text object, or of all text for a plain text object.

- `setFont:` (page 32)

  Sets the font of all the receiver's text to *`aFont`*.

- `setFont:range:` (page 33)

  Sets the font of characters within *`aRange`* to *`aFont`*.

- `font` (page 20)

  Returns the font of the first character in the receiver's text, or of the insertion point if there's no text.

Setting Text Alignment

- `setAlignment:` (page 29)

  Sets the alignment of all the receiver's text to *`mode`*.

- `alignCenter:` (page 14)

  This action method applies center alignment to selected paragraphs (or all text if the receiver is a plain text object).

- `alignLeft:` (page 14)

  This action method applies left alignment to selected paragraphs (or all text if the receiver is a plain text object).

- `alignRight:` (page 15)

  This action method applies right alignment to selected paragraphs (or all text if the receiver is a plain text object).

- `alignment` (page 15)

  Returns the alignment of the first paragraph (or all text if the receiver is a plain text object).

Setting Text Color

- `setTextColor:` (page 36)

  Sets the text color of all characters in the receiver to *`aColor`*.

- `setTextColor:range:` (page 37)

  Sets the text color of characters within *`aRange`* to *`aColor`*.

- `textColor` (page 40)
  Returns the color of the receiver's first character, or for the insertion point if there's no text.

Writing Direction

- `baseWritingDirection` (page 16)
  Returns the initial writing direction used to determine the actual writing direction for text.
- `setBaseWritingDirection:` (page 30)
  Sets the initial writing direction used to determine the actual writing direction for text.

Setting Superscripting and Subscripting

- `superscript:` (page 40)
  This action method applies a superscript attribute to selected text (or all text if the receiver is a plain text object), raising its baseline offset by a predefined amount.
- `subscript:` (page 40)
  This action method applies a subscript attribute to selected text (or all text if the receiver is a plain text object), lowering its baseline offset by a predefined amount.
- `unscript:` (page 41)
  This action method removes any superscripting or subscripting from selected text (or all text if the receiver is a plain text object).

Underlining Text

- `underline:` (page 41)
  Adds the underline attribute to the selected text attributes if absent; removes the attribute if present.

Reading and Writing RTF Files

- `readRTFDFromFile:` (page 25)
  Attempts to read the RTFD file at *path*, returning `YES` if successful and `NO` if not.
- `writeRTFDToFile:atomically:` (page 42)
  Writes the receiver's text as RTF with attachments to a file or directory at *path*.
- `RTFDFromRange:` (page 27)
  Returns an NSData object that contains an RTFD stream corresponding to the characters and attributes within *aRange*.
- `RTFFromRange:` (page 28)
  Returns an NSData object that contains an RTF stream corresponding to the characters and attributes within *aRange*, omitting any attachment characters and attributes.

Checking Spelling

- `checkSpelling:` (page 17)
  This action method searches for a misspelled word in the receiver's text.
- `showGuessPanel:` (page 38)
  This action method opens the Spelling panel, allowing the user to make a correction during spell checking.

Constraining Size

- `setMaxSize:` (page 34)
  Sets the receiver's maximum size to *aSize*.
- `maxSize` (page 23)
  Returns the receiver's maximum size.
- `setMinSize:` (page 34)
  Sets the receiver's minimum size to *aSize*.
- `minSize` (page 24)
  Returns the receiver's minimum size.
- `setVerticallyResizable:` (page 38)
  Controls whether the receiver changes its height to fit the height of its text.
- `isVerticallyResizable` (page 23)
  Returns `YES` if the receiver automatically changes its height to accommodate the height of its text, `NO` if it doesn't.
- `setHorizontallyResizable:` (page 33)
  Controls whether the receiver changes its width to fit the width of its text.
- `isHorizontallyResizable` (page 21)
  Returns `YES` if the receiver automatically changes its width to accommodate the width of its text, `NO` if it doesn't.
- `sizeToFit` (page 39)
  Resizes the receiver to fit its text.

Scrolling

- `scrollRangeToVisible:` (page 28)
  Scrolls the receiver in its enclosing scroll view so the first characters of *aRange* are visible.

Setting the Delegate

- `setDelegate:` (page 30)
  Sets the receiver's delegate.
- `delegate` (page 19)
  Returns the receiver's delegate.

Editing text

- textShouldBeginEditing: (page 43) *delegate method*
  Invoked when a text object begins to change its text, this method requests permission for *aTextObject* to begin editing.
- textDidBeginEditing: (page 42) *delegate method*
  Informs the delegate that the text object has begun editing (that the user has begun changing it).
- textShouldEndEditing: (page 44) *delegate method*
  Invoked from a text object's implementation of `resignFirstResponder`, this method requests permission for *aTextObject* to end editing.
- textDidEndEditing: (page 43) *delegate method*
  Informs the delegate that the text object has finished editing (that it has resigned first responder status).

Changing text formatting

- textDidChange: (page 43) *delegate method*
  Informs the delegate that the text object has changed its characters or formatting attributes.

Instance Methods

alignCenter:

This action method applies center alignment to selected paragraphs (or all text if the receiver is a plain text object).

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

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

See Also
- alignLeft: (page 14)
- alignRight: (page 15)
- alignment (page 15)
- setAlignment: (page 29)

Declared In
`NSText.h`

alignLeft:

This action method applies left alignment to selected paragraphs (or all text if the receiver is a plain text object).

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

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

See Also
- `alignCenter:` (page 14)
- `alignRight:` (page 15)
- `alignment` (page 15)
- `setAlignment:` (page 29)

Declared In
`NSText.h`

alignment

Returns the alignment of the first paragraph (or all text if the receiver is a plain text object).

```
- (NSTextAlignment)alignment
```

Discussion
The returned value is one of the alignments described in NSTextAlignment (page 44).

Text using `NSNaturalTextAlignment` is actually displayed using one of the other alignments, depending on the natural alignment of the text's script.

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

Declared In
`NSText.h`

alignRight:

This action method applies right alignment to selected paragraphs (or all text if the receiver is a plain text object).

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

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

See Also
- `alignLeft:` (page 14)
- `alignCenter:` (page 14)
- `alignment` (page 15)
- `setAlignment:` (page 29)

Declared In
`NSText.h`

backgroundColor

Returns the receiver's background color.

```
- (NSColor *)backgroundColor
```

Return Value

The receiver's background color.

Availability

Available in Mac OS X v10.0 and later.

See Also

- drawsBackground (page 19)
- setBackgroundColor: (page 30)

Declared In

NSText.h

baseWritingDirection

Returns the initial writing direction used to determine the actual writing direction for text.

```
- (NSWritingDirection)baseWritingDirection
```

Discussion

The Text system uses this value as a hint for calculating the actual direction for displaying Unicode characters. You should not need to call this method directly. If no writing direction is set, returns NSWritingDirectionNatural.

Availability

Available in Mac OS X v10.4 and later.

See Also

- setBaseWritingDirection: (page 30)

Declared In

NSText.h

changeFont:

This action method changes the font of the selection for a rich text object, or of all text for a plain text object.

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

Discussion

If the receiver doesn't use the Font panel, this method does nothing.

This method changes the font by sending a convertFont: message to the shared NSFontManager and applying each NSFont returned to the appropriate text. See the NSFontManager class specification for more information on font conversion.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `usesFontPanel` (page 42)

Declared In
`NSText.h`

checkSpelling:

This action method searches for a misspelled word in the receiver's text.

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

Discussion
The search starts at the end of the selection and continues until it reaches a word suspected of being misspelled or the end of the text. If a word isn't recognized by the spelling server, a `showGuessPanel:` (page 38) message then opens the Guess panel and allows the user to make a correction or add the word to the local dictionary.

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

See Also
- `showGuessPanel:` (page 38)

Declared In
`NSText.h`

copy:

This action method copies the selected text onto the general pasteboard, in as many formats as the receiver supports.

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

Discussion
A plain text object uses `NSStringPboardType` for plain text, and a rich text object also uses `NSRTFPboardType`.

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

See Also
- `copyFont:` (page 18)
- `copyRuler:` (page 18)
- `cut:` (page 18)
- `paste:` (page 24)

Declared In
`NSText.h`

copyFont:

This action method copies the font information for the first character of the selection (or for the insertion point) onto the font pasteboard, as `NSFontPboardType`.

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

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

See Also
`– copy:` (page 17)
`– copyRuler:` (page 18)
`– cut:` (page 18)
`– paste:` (page 24)

Declared In
`NSText.h`

copyRuler:

This action method copies the paragraph style information for first selected paragraph onto the ruler pasteboard, as `NSRulerPboardType`, and expands the selection to paragraph boundaries.

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

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

See Also
`– copy:` (page 17)
`– copyFont:` (page 18)
`– cut:` (page 18)
`– paste:` (page 24)

Declared In
`NSText.h`

cut:

This action method deletes the selected text and places it onto the general pasteboard, in as many formats as the receiver supports.

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

Discussion
A plain text object uses `NSStringPboardType` for plain text, and a rich text object also uses `NSRTFPboardType`.

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

See Also

- `- delete:` (page 19)
- `- copy:` (page 17)
- `- copyFont:` (page 18)
- `- copyRuler:` (page 18)
- `- paste:` (page 24)

Declared In

`NSText.h`

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Return Value

The receiver's delegate, or `nil` if it has none.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `- setDelegate:` (page 30)

Declared In

`NSText.h`

delete:

This action method deletes the selected text.

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

Availability

Available in Mac OS X v10.0 and later.

See Also

- `- cut:` (page 18)

Declared In

`NSText.h`

drawsBackground

Returns a Boolean value that indicates whether the receiver draws its background.

```
- (BOOL)drawsBackground
```

Return Value

`YES` if the receiver draws its background, otherwise `NO`.

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

See Also
- backgroundColor (page 16)
- setDrawsBackground: (page 31)

Declared In
`NSText.h`

font

Returns the font of the first character in the receiver's text, or of the insertion point if there's no text.

```
- (NSFont *)font
```

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

See Also
- setFont: (page 32)
- setFont:range: (page 33)

Related Sample Code
Aperture Edit Plugin - Borders & Titles
TipWrapper

Declared In
`NSText.h`

importsGraphics

Returns a Boolean value that indicates whether the receiver allows the user to import files by dragging.

```
- (BOOL)importsGraphics
```

Return Value
`YES` if the receiver allows the user to import files by dragging, otherwise `NO`.

Discussion
A text object that accepts dragged files is also a rich text object.

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

See Also
- isRichText (page 22)
- setImportsGraphics: (page 33)

Declared In
`NSText.h`

isEditable

Returns a Boolean value that indicates whether the receiver allows the user to edit text, NO if it doesn't.

```
- (BOOL)isEditable
```

Return Value

YES if the receiver allows the user to edit text, otherwise NO.

Discussion

You can change the receiver's text programmatically regardless of this setting.

If the receiver is editable, it's also selectable.

Availability

Available in Mac OS X v10.0 and later.

See Also

- isSelectable (page 23)
- setEditable: (page 31)

Declared In

NSText.h

isFieldEditor

Returns a Boolean value that indicates whether the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder.

```
- (BOOL)isFieldEditor
```

Return Value

YES if the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder; NO if it accepts them as text input.

Discussion

See the NSWindow class specification for more information on field editors. By default, NSText objects don't behave as field editors.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setFieldEditor: (page 32)

Declared In

NSText.h

isHorizontallyResizable

Returns YES if the receiver automatically changes its width to accommodate the width of its text, NO if it doesn't.

```
- (BOOL)isHorizontallyResizable
```

Discussion
By default, an `NSText` object is not horizontally resizable.

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

See Also
- `isVerticallyResizable` (page 23)
- `setHorizontallyResizable:` (page 33)

Related Sample Code
Sketch-112

Declared In
`NSText.h`

isRichText

Returns a Boolean value that indicates whether the receiver allows the user to apply attributes to specific ranges of the text.

```
- (BOOL)isRichText
```

Return Value
`YES` if the receiver allows the user to apply attributes to specific ranges of the text, otherwise `NO`.

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

See Also
- `importsGraphics` (page 20)
- `setRichText:` (page 35)

Declared In
`NSText.h`

isRulerVisible

Returns a Boolean value that indicates whether the receiver's enclosing scroll view shows its ruler.

```
- (BOOL)isRulerVisible
```

Return Value
`YES` if the receiver's enclosing scroll view shows its ruler, otherwise `NO`.

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

See Also
- `toggleRuler:` (page 41)

Declared In
`NSText.h`

isSelectable

Returns a Boolean value that indicates whether the receiver allows the user to select text, `NO` if it doesn’t.

```
- (BOOL)isSelectable
```

Return Value

`YES` if the receiver allows the user to select text, otherwise `NO`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- isEditable` (page 21)
`- setSelectable:` (page 35)

Declared In

`NSText.h`

isVerticallyResizable

Returns `YES` if the receiver automatically changes its height to accommodate the height of its text, `NO` if it doesn’t.

```
- (BOOL)isVerticallyResizable
```

Discussion

By default, an `NSText` object is vertically resizable.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- isHorizontallyResizable` (page 21)
`- setVerticallyResizable:` (page 38)

Declared In

`NSText.h`

maxSize

Returns the receiver’s maximum size.

```
- (NSSize)maxSize
```

Availability

Available in Mac OS X v10.0 and later.

See Also

`- minSize` (page 24)
`- setMaxSize:` (page 34)

Declared In

`NSText.h`

minSize

Returns the receiver's minimum size.

```
- (NSSize)minSize
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- `– maxSize` (page 23)
- `– setMinSize:` (page 34)

Declared In

`NSText.h`

paste:

This action method pastes text from the general pasteboard at the insertion point or over the selection.

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

Availability

Available in Mac OS X v10.0 and later.

See Also

- `– copy:` (page 17)
- `– cut:` (page 18)
- `– pasteFont:` (page 24)
- `– pasteRuler:` (page 25)

Declared In

`NSText.h`

pasteFont:

This action method pastes font information from the font pasteboard onto the selected text or insertion point of a rich text object, or over all text of a plain text object.

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

Availability

Available in Mac OS X v10.0 and later.

See Also

- `– copyFont:` (page 18)
- `– pasteRuler:` (page 25)

Declared In

`NSText.h`

pasteRuler:

This action method pastes paragraph style information from the ruler pasteboard onto the selected paragraphs of a rich text object.

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

Discussion

It doesn't apply to a plain text object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `copyFont:` (page 18)
- `pasteRuler:` (page 25)

Declared In

`NSText.h`

readRTFDFromFile:

Attempts to read the RTFD file at *path*, returning `YES` if successful and `NO` if not.

```
- (BOOL)readRTFDFromFile:(NSString *)path
```

Discussion

path should be the path for an `.rtf` file or an `.rtfd` file wrapper, not for the RTF file within an `.rtfd` file wrapper.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `writeRTFDToFile:atomically:` (page 42)

Declared In

`NSText.h`

replaceCharactersInRange:withRTF:

Replaces the characters in the given range with RTF text interpreted from the given RTF data.

```
- (void)replaceCharactersInRange:(NSRange)aRange withRTF:(NSData *)rtfData
```

Parameters

aRange

The range of characters to be replaced.

rtfData

The RTF data from which to derive the replacement string.

Discussion

This method applies only to rich text objects.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` or `shouldChangeTextInRange:replacementString:` to include this method in an undoable action.

This method is designed for transferring text from out-of-process sources such as the pasteboard. In most cases, programmatic modification of the text is best done by operating on the text storage directly, using the general methods of `NSMutableAttributedString`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- replaceCharactersInRange:withRTFD:` (page 26)
`- replaceCharactersInRange:withString:` (page 27)

Declared In

`NSText.h`

replaceCharactersInRange:withRTFD:

Replaces the characters in the given range with RTFD text interpreted from the given RTFD data.

```
- (void)replaceCharactersInRange:(NSRange)aRange withRTFD:(NSData *)rtfdData
```

Parameters

aRange

The range of characters to be replaced.

rtfdData

The RTFD data from which to derive the replacement string.

Discussion

This method applies only to rich text objects.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` or `shouldChangeTextInRange:replacementString:` to include this method in an undoable action.

This method is designed for transferring text from out-of-process sources such as the pasteboard. In most cases, programmatic modification of the text is best done by operating on the text storage directly, using the general methods of `NSMutableAttributedString`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- replaceCharactersInRange:withRTF:` (page 25)
`- replaceCharactersInRange:withString:` (page 27)

Declared In

`NSText.h`

replaceCharactersInRange:withString:

Replaces the characters in the given range with those in the given string.

```
- (void)replaceCharactersInRange:(NSRange)aRange withString:(NSString *)aString
```

Parameters

aRange

The range of characters to be replaced.

aString

The replacement string.

Discussion

For a rich text object, the text of *aString* is assigned the formatting attributes of the first character of the text it replaces, or of the character immediately before *aRange* if the range's length is 0. If the range's location is 0, the formatting attributes of the first character in the receiver are used.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` or `shouldChangeTextInRange:replacementString:` to include this method in an undoable action.

In most cases, programmatic modification of the text is best done by operating on the text storage directly, using the general methods of `NSMutableAttributedString`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `- replaceCharactersInRange:withRTF:` (page 25)
- `- replaceCharactersInRange:withRTFD:` (page 26)

Declared In

`NSText.h`

RTFDFromRange:

Returns an NSData object that contains an RTFD stream corresponding to the characters and attributes within *aRange*.

```
- (NSData *)RTFDFromRange:(NSRange)aRange
```

Discussion

Raises an `NSRangeException` if any part of *aRange* lies beyond the end of the receiver's characters.

When writing data to the pasteboard, you can use the NSData object as the first argument to `NSPasteboard`'s `setData:forType:` method, with a second argument of `NSRTFDPboardType`.

Availability

Available in Mac OS X v10.0 and later.

See Also
`- RTFFromRange:` (page 28)

Declared In
`NSText.h`

RTFFromRange:

Returns an NSData object that contains an RTF stream corresponding to the characters and attributes within *aRange*, omitting any attachment characters and attributes.

```
- (NSData *)RTFFromRange:(NSRange)aRange
```

Discussion
Raises an `NSRangeException` if any part of *aRange* lies beyond the end of the receiver's characters.

When writing data to the pasteboard, you can use the NSData object as the first argument to `NSPasteboard`'s `setData:forType:` method, with a second argument of `NSRTFPboardType`.

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

See Also
`- RTFDFromRange:` (page 27)

Declared In
`NSText.h`

scrollRangeToVisible:

Scrolls the receiver in its enclosing scroll view so the first characters of *aRange* are visible.

```
- (void)scrollRangeToVisible:(NSRange)aRange
```

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

Related Sample Code
BackgroundExporter

Declared In
`NSText.h`

selectAll:

This action method selects all of the receiver's text.

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

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

Related Sample Code
Aperture Edit Plugin - Borders & Titles

Declared In
`NSText.h`

selectedRange

Returns the range of selected characters.

```
- (NSRange)selectedRange
```

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

See Also
`- setSelectedRange:` (page 36)

Related Sample Code
TextLinks

Declared In
`NSText.h`

setAlignment:

Sets the alignment of all the receiver's text to *`mode`*.

```
- (void)setAlignment:(NSTextAlignment)mode
```

Discussion
The value of *`mode`* must be one of the alignments described in NSTextAlignment (page 44).

Text using `NSNaturalTextAlignment` is actually displayed using one of the other alignments, depending on the natural alignment of the text's script.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` or `shouldChangeTextInRange:replacementString:` to include this method in an undoable action.

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

See Also
`- alignment` (page 15)
`- alignLeft:` (page 14)
`- alignCenter:` (page 14)
`- alignRight:` (page 15)

Declared In
`NSText.h`

setBackgroundColor:

Sets the receiver's background color to a given color.

```
- (void)setBackgroundColor:(NSColor *)aColor
```

Parameters

aColor

The background color for the receiver.

Discussion

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` or `shouldChangeTextInRange:replacementString:` to include this method in an undoable action.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setDrawsBackground:` (page 31)
- `backgroundColor` (page 16)

Declared In

`NSText.h`

setBaseWritingDirection:

Sets the initial writing direction used to determine the actual writing direction for text.

```
- (void)setBaseWritingDirection:(NSWritingDirection)writingDirection
```

Discussion

If you know the base writing direction of the text you are rendering, you can use this method to specify that direction to the text system.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` or `shouldChangeTextInRange:replacementString:` to include this method in an undoable action.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `baseWritingDirection` (page 16)

Declared In

`NSText.h`

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)anObject
```

Parameters

anObject

The delegate for the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `delegate` (page 19)

Declared In

`NSText.h`

setDrawsBackground:

Controls whether the receiver draws its background.

```
- (void)setDrawsBackground:(BOOL)flag
```

Parameters

flag

If *flag* is `YES`, the receiver fills its background with the background color, if *flag* is `NO`, it doesn't.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setBackgroundColor:` (page 30)
- `drawsBackground` (page 19)

Declared In

`NSText.h`

setEditable:

Controls whether the receiver allows the user to edit its text.

```
- (void)setEditable:(BOOL)flag
```

Parameters

flag

If *flag* is `YES`, the receiver allows the user to edit text and attributes; if *flag* is `NO`, it doesn't.

Discussion

You can change the receiver's text programmatically regardless of this setting. If the receiver is made editable, it's also made selectable. `NSText` objects are by default editable.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setSelectable:` (page 35)
- `isEditable` (page 21)

Declared In
`NSText.h`

setFieldEditor:

Controls whether the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder.

```
- (void)setFieldEditor:(BOOL)flag
```

Parameters

flag

If *flag* is `YES`, the receiver interprets Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder; if *flag* is `NO`, it doesn't, instead accepting these characters as text input.

Discussion
See the NSWindow class specification for more information on field editors. By default, `NSText` objects don't behave as field editors.

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

See Also
`- isFieldEditor` (page 21)

Declared In
`NSText.h`

setFont:

Sets the font of all the receiver's text to *aFont*.

```
- (void)setFont:(NSFont *)aFont
```

Discussion
This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` or `shouldChangeTextInRange:replacementString:` to include this method in an undoable action.

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

See Also
`- setFont:range:` (page 33)
`- font` (page 20)

Related Sample Code
TipWrapper

Declared In
`NSText.h`

setFont:range:

Sets the font of characters within *aRange* to *aFont*.

```
- (void)setFont:(NSFont *)aFont range:(NSRange)aRange
```

Discussion
This method applies only to a rich text object.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` or `shouldChangeTextInRange:replacementString:` to include this method in an undoable action.

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

See Also
- `setFont:` (page 32)
- `font` (page 20)

Declared In
`NSText.h`

setHorizontallyResizable:

Controls whether the receiver changes its width to fit the width of its text.

```
- (void)setHorizontallyResizable:(BOOL)flag
```

Discussion
If *flag* is `YES` it does; if *flag* is `NO` it doesn't.

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

See Also
- `setVerticallyResizable:` (page 38)
- `isHorizontallyResizable` (page 21)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

Sketch-112

TextEditPlus

TipWrapper

TrackBall

Declared In
`NSText.h`

setImportsGraphics:

Controls whether the receiver allows the user to import files by dragging.

```
- (void)setImportsGraphics:(BOOL)flag
```

Parameters

flag

If *flag* is YES, the receiver allows the user to import files by dragging; if *flag* is NO, it doesn't.

Discussion

If the receiver is set to accept dragged files, it's also made a rich text object. Subclasses may or may not accept dragged files by default.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setRichText: (page 35)
- importsGraphics (page 20)

Declared In

NSText.h

setMaxSize:

Sets the receiver's maximum size to *aSize*.

```
- (void)setMaxSize:(NSSize)aSize
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- setMinSize: (page 34)
- maxSize (page 23)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Sketch-112

TextEditPlus

TipWrapper

Declared In

NSText.h

setMinSize:

Sets the receiver's minimum size to *aSize*.

```
- (void)setMinSize:(NSSize)aSize
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setMaxSize:` (page 34)
- `minSize` (page 24)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Sketch-112

TextEditPlus

TipWrapper

Declared In

`NSText.h`

setRichText:

Controls whether the receiver allows the user to apply attributes to specific ranges of the text.

```
- (void)setRichText:(BOOL)flag
```

Parameters

flag

If *flag* is `YES` the receiver allows the user to apply attributes to specific ranges of the text; if *flag* is `NO` it doesn't.

Discussion

If *flag* is `NO`, the receiver is also set not to accept dragged files. Subclasses may or may not let the user apply multiple attributes to the text and accept drag files by default.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `isRichText` (page 22)
- `setImportsGraphics:` (page 33)

Declared In

`NSText.h`

setSelectable:

Controls whether the receiver allows the user to select its text.

```
- (void)setSelectable:(BOOL)flag
```

Parameters

flag

If *flag* is `YES`, the receiver allows the user to select text; if *flag* is `NO`, it doesn't.

Discussion

You can set selections programmatically regardless of this setting. If the receiver is made not selectable, it's also made not editable. `NSText` objects are by default editable and selectable.

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

See Also
- `setEditable:` (page 31)
- `isSelectable` (page 23)

Declared In
`NSText.h`

setSelectedRange:

Selects the receiver's characters within *aRange*.

```
- (void)setSelectedRange:(NSRange)aRange
```

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

See Also
- `selectedRange` (page 29)

Declared In
`NSText.h`

setString:

Replaces the receiver's entire text with *aString*, applying the formatting attributes of the old first character to its new contents.

```
- (void)setString:(NSString *)aString
```

Discussion
This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` or `shouldChangeTextInRange:replacementString:` to include this method in an undoable action.

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

Related Sample Code
TipWrapper

Declared In
`NSText.h`

setTextColor:

Sets the text color of all characters in the receiver to *aColor*.

```
- (void)setTextColor:(NSColor *)aColor
```

Discussion

Removes the text color attribute if *aColor* is `nil`.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` or `shouldChangeTextInRange:replacementString:` to include this method in an undoable action.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setTextColor:range:` (page 37)
- `textColor` (page 40)

Declared In

`NSText.h`

setTextColor:range:

Sets the text color of characters within *aRange* to *aColor*.

```
- (void)setTextColor:(NSColor *)aColor range:(NSRange)aRange
```

Discussion

Removes the text color attribute if *aColor* is `nil`. This method applies only to rich text objects.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` or `shouldChangeTextInRange:replacementString:` to include this method in an undoable action.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setTextColor:` (page 36)
- `textColor` (page 40)

Related Sample Code

TextViewDelegate

Declared In

`NSText.h`

setUsesFontPanel:

Controls whether the receiver uses the Font panel and Font menu.

```
- (void)setUsesFontPanel:(BOOL)flag
```

Parameters

flag

If *flag* is YES, the receiver responds to messages from the Font panel and from the Font menu and updates the Font panel with the selection font whenever it changes. If *flag* is NO the receiver doesn't do any of these actions.

Discussion

By default, an NSText object uses the Font panel and menu.

Availability

Available in Mac OS X v10.0 and later.

See Also

- usesFontPanel (page 42)

Declared In

NSText.h

setVerticallyResizable:

Controls whether the receiver changes its height to fit the height of its text.

```
- (void)setVerticallyResizable:(BOOL)flag
```

Discussion

If *flag* is YES it does; if *flag* is NO it doesn't.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setHorizontallyResizable: (page 33)
- isVerticallyResizable (page 23)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Sketch-112

TextEditPlus

TipWrapper

Declared In

NSText.h

showGuessPanel:

This action method opens the Spelling panel, allowing the user to make a correction during spell checking.

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

Availability

Available in Mac OS X v10.0 and later.

See Also

- `checkSpelling:` (page 17)

Declared In

`NSText.h`

sizeToFit

Resizes the receiver to fit its text.

```
- (void)sizeToFit
```

Discussion

The text view will not be sized any smaller than its minimum size, however.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `isHorizontallyResizable` (page 21)
- `isVerticallyResizable` (page 23)

Related Sample Code

TipWrapper

Declared In

`NSText.h`

string

Returns the characters of the receiver's text.

```
- (NSString *)string
```

Return Value

The characters of the receiver's text.

Discussion

For performance reasons, this method returns the current backing store of the text object. If you want to maintain a snapshot of this as you manipulate the text storage, you should make a copy of the appropriate substring.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setString:` (page 36)

Related Sample Code

SearchField

TextViewDelegate

TrackBall

Declared In
`NSText.h`

subscript:

This action method applies a subscript attribute to selected text (or all text if the receiver is a plain text object), lowering its baseline offset by a predefined amount.

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

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

See Also
`– subscript:` (page 40)
`– unscript:` (page 41)
`– lowerBaseline:` (NSTextView)

Declared In
`NSText.h`

superscript:

This action method applies a superscript attribute to selected text (or all text if the receiver is a plain text object), raising its baseline offset by a predefined amount.

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

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

See Also
`– subscript:` (page 40)
`– unscript:` (page 41)
`– raiseBaseline:` (NSTextView)

Declared In
`NSText.h`

textColor

Returns the color of the receiver's first character, or for the insertion point if there's no text.

```
- (NSColor *)textColor
```

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

See Also
`– setTextColor:` (page 36)
`– setTextColor:range:` (page 37)

Declared In

`NSText.h`

toggleRuler:

This action method shows or hides the ruler, if the receiver is enclosed in a scroll view.

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

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSText.h`

underline:

Adds the underline attribute to the selected text attributes if absent; removes the attribute if present.

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

Discussion

If there is a selection and the first character of the selected range has any form of underline on it, or if there is no selection and the typing attributes have any form of underline, then underline is removed; otherwise a single simple underline is added.

Operates on the selected range if the receiver contains rich text. For plain text the range is the entire contents of the receiver.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSText.h`

unscript:

This action method removes any superscripting or subscripting from selected text (or all text if the receiver is a plain text object).

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

Availability

Available in Mac OS X v10.0 and later.

See Also

`- subscript:` (page 40)
`- superscript:` (page 40)
`- raiseBaseline:` (NSTextView)
`- lowerBaseline:` (NSTextView)

Declared In
`NSText.h`

usesFontPanel

Returns a Boolean value that indicates whether the receiver uses the Font panel.

```
- (BOOL)usesFontPanel
```

Return Value
`YES` if the receiver uses the Font panel, otherwise `NO`.

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

See Also
`- setUsesFontPanel:` (page 37)

Declared In
`NSText.h`

writeRTFDToFile:atomically:

Writes the receiver's text as RTF with attachments to a file or directory at *`path`*.

```
- (BOOL)writeRTFDToFile:(NSString *)path atomically:(BOOL)atomicFlag
```

Discussion
Returns `YES` on success and `NO` on failure. If *`atomicFlag`* is `YES`, attempts to write the file safely so that an existing file at *`path`* is not overwritten, nor does a new file at *`path`* actually get created, unless the write is successful.

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

See Also
`- RTFFromRange:` (page 28)
`- RTFDFromRange:` (page 27)
`- readRTFDFromFile:` (page 25)

Declared In
`NSText.h`

Delegate Methods

textDidBeginEditing:

Informs the delegate that the text object has begun editing (that the user has begun changing it).

```
- (void)textDidBeginEditing:(NSNotification *)aNotification
```

Discussion
The name of *aNotification* is `NSTextDidBeginEditingNotification` (page 48).

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

Declared In
`NSText.h`

textDidChange:

Informs the delegate that the text object has changed its characters or formatting attributes.

```
- (void)textDidChange:(NSNotification *)aNotification
```

Discussion
The name of *aNotification* is `NSTextDidChangeNotification` (page 49).

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

Declared In
`NSText.h`

textDidEndEditing:

Informs the delegate that the text object has finished editing (that it has resigned first responder status).

```
- (void)textDidEndEditing:(NSNotification *)aNotification
```

Discussion
The name of *aNotification* is `NSTextDidEndEditingNotification` (page 49).

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

Declared In
`NSText.h`

textShouldBeginEditing:

Invoked when a text object begins to change its text, this method requests permission for *aTextObject* to begin editing.

```
- (BOOL)textShouldBeginEditing:(NSText *)aTextObject
```

Discussion
If the delegate returns `YES`, the text object proceeds to make changes. If the delegate returns `NO`, the text object abandons the editing operation. This method is also invoked when the user drags and drops a file onto the text object.

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

See Also
- `makeFirstResponder:` (NSWindow)
- `becomeFirstResponder` (NSResponder)

Declared In
`NSText.h`

textShouldEndEditing:

Invoked from a text object's implementation of `resignFirstResponder`, this method requests permission for *`aTextObject`* to end editing.

```
- (BOOL)textShouldEndEditing:(NSText *)aTextObject
```

Discussion
If the delegate returns `YES`, the text object proceeds to finish editing and resign first responder status. If the delegate returns `NO`, the text object selects all of its text and remains the first responder.

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

See Also
- `resignFirstResponder` (NSResponder)

Declared In
`NSText.h`

Constants

NSTextAlignment

These constants specify text alignment.

```
typedef enum _NSTextAlignment {
    NSLeftTextAlignment      = 0,
    NSRightTextAlignment     = 1,
    NSCenterTextAlignment    = 2,
    NSJustifiedTextAlignment = 3,
    NSNaturalTextAlignment   = 4
} NSTextAlignment;
```

Constants
`NSLeftTextAlignment`

Text is visually left aligned.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSRightTextAlignment`
: Text is visually right aligned.

  Available in Mac OS X v10.0 and later.

  Declared in `NSText.h`.

`NSCenterTextAlignment`
: Text is visually center aligned.

  Available in Mac OS X v10.0 and later.

  Declared in `NSText.h`.

`NSJustifiedTextAlignment`
: Text is justified.

  Available in Mac OS X v10.0 and later.

  Declared in `NSText.h`.

`NSNaturalTextAlignment`
: Use the natural alignment of the text's script.

  Available in Mac OS X v10.0 and later.

  Declared in `NSText.h`.

Declared In
`NSText.h`

NSWritingDirection

These constants specify the writing directions:

```
typedef enum _NSWritingDirection {
    NSWritingDirectionNatural     = -1,
    NSWritingDirectionLeftToRight = 0,
    NSWritingDirectionRightToLeft
} NSWritingDirection;
```

Constants
`NSWritingDirectionNatural`
: The writing direction is determined using the Unicode Bidi Algorithm rules P2 and P3. Default.

  Available in Mac OS X v10.4 and later.

  Declared in `NSText.h`.

`NSWritingDirectionLeftToRight`
: The writing direction is left to right.

  Available in Mac OS X v10.2 and later.

  Declared in `NSText.h`.

`NSWritingDirectionRightToLeft`
: The writing direction is right to left.

  Available in Mac OS X v10.2 and later.

  Declared in `NSText.h`.

Declared In
`NSText.h`

Movement Codes

These constants specify the reason for a change of editing focus among text fields, in essence answering the question “why am I leaving the field?”

```
enum {
   NSIllegalTextMovement = 0,
   NSReturnTextMovement  = 0x10,
   NSTabTextMovement     = 0x11,
   NSBacktabTextMovement = 0x12,
   NSLeftTextMovement    = 0x13,
   NSRightTextMovement   = 0x14,
   NSUpTextMovement      = 0x15,
   NSDownTextMovement    = 0x16,
   NSCancelTextMovement  = 0x17,
  NSOtherTextMovement    = 0
};
```

Constants

`NSIllegalTextMovement`

Currently unused.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSReturnTextMovement`

The Return key was pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSTabTextMovement`

The Tab key was pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSBacktabTextMovement`

The Backtab (Shift-Tab) key was pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSLeftTextMovement`

The left arrow key was pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSRightTextMovement`

The right arrow key was pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSUpTextMovement`

The up arrow key was pressed.

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSDownTextMovement`
    The down arrow key was pressed.

    Available in Mac OS X v10.0 and later.

    Declared in `NSText.h`.

`NSCancelTextMovement`
    The user cancelled the completion.

    Available in Mac OS X v10.3 and later.

    Declared in `NSText.h`.

`NSOtherTextMovement`
    The user performed some undefined action.

    Available in Mac OS X v10.3 and later.

    Declared in `NSText.h`.

Discussion

They are the possible values for the `NSTextMovement` key of the `NSTextDidEndEditingNotification` (page 49)`userInfo` dictionary. The field editor makes sure that these are the values sent when the user presses the Tab, Backtab, or Return key while editing. The control then uses this information to decide where to send focus next.

Declared In

`NSText.h`

Commonly-used Unicode characters

These constants specify several commonly used Unicode characters.

```
enum {
   NSParagraphSeparatorCharacter = 0x2029,
   NSLineSeparatorCharacter      = 0x2028,
   NSTabCharacter                = 0x0009,
   NSFormFeedCharacter           = 0x000c,
   NSNewlineCharacter            = 0x000a,
   NSCarriageReturnCharacter     = 0x000d,
   NSEnterCharacter              = 0x0003,
   NSBackspaceCharacter          = 0x0008,
   NSBackTabCharacter            = 0x0019,
   NSDeleteCharacter             = 0x007f
};
```

Constants

`NSParagraphSeparatorCharacter`
    The paragraph separator character: `0x2029`

    Available in Mac OS X v10.0 and later.

    Declared in `NSText.h`.

`NSLineSeparatorCharacter`
    The line separator character: `0x2028`

    Available in Mac OS X v10.0 and later.

    Declared in `NSText.h`.

`NSTabCharacter`

The tab character: `0x0009`

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSBackTabCharacter`

The back tab character: `0x0019`

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSFormFeedCharacter`

The form feed character: `0x000c`

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSNewlineCharacter`

The newline character: `0x000a`

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSCarriageReturnCharacter`

The carriage return character: `0x000d`

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSEnterCharacter`

The enter character: `0x0003`

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSBackspaceCharacter`

The backspace character: `0x0008`

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

`NSDeleteCharacter`

The delete character: `0x007f`

Available in Mac OS X v10.0 and later.

Declared in `NSText.h`.

Declared In

`NSText.h`

Notifications

NSTextDidBeginEditingNotification

Posted when an `NSText` object begins any operation that changes characters or formatting attributes.

The notification object is the notifying `NSText` object. This notification does not contain a *`userInfo`* dictionary.

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

Declared In
`NSText.h`

NSTextDidChangeNotification

Posted after an `NSText` object performs any operation that changes characters or formatting attributes.

The notification object is the notifying `NSText` object. This notification does not contain a *`userInfo`* dictionary.

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

Declared In
`NSText.h`

NSTextDidEndEditingNotification

Posted when focus leaves an `NSText` object, whether or not any operation has changed characters or formatting attributes.

The notification object is the notifying `NSText` object. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSTextMovement"`	Possible movement code values are described in Movement Codes (page 46).

Note: It is common for `NSTextDidEndEditingNotification` to be sent without a matching `NSTextDidBeginEditingNotification`. The begin notification is only sent if the user actually makes changes (that is, types something or changes formatting attributes). However, the end notification is sent when focus leaves the text view, regardless of whether there was a change.

This distinction enables an application to know whether the user actually made a change to the text or just clicked in the text view and then clicked outside it. In both cases, `NSTextDidEndEditingNotification` is sent, but to tell the difference, the application can listen for `NSTextDidBeginEditingNotification`.

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

Declared In
`NSText.h`

Document Revision History

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

Date	Notes
2008-05-06	Added a note explaining why NSTextDidEndEditingNotification may be sent without a matching NSTextDidBeginEditingNotification.
2007-12-11	Clarified description of alignment method.
2007-03-24	Added guidance to replaceCharactersInRange... methods that they are intended for transferring text from extra-process resources, such as the pasteboard.
	Changed `textShouldBeginEditing:` (page 43) description to indicate it is invoked when the text is changed, rather than in the text object's `becomeFirstResponder` implementation.
	Amplified description of `underline:` (page 41).
2006-06-28	Made minor changes to conform to reference consistency guidelines.
2006-05-23	First publication of this content as a separate document.

Index