NSTextView Class Reference

Cocoa > Text & Fonts

2007-04-27

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

Simultaneously published in the United States and Canada.

Contents

NSTextView Class Reference

Inherits from	NSText : NSView : NSResponder : NSObject
Conforms to	NSTextInput NSUserInterfaceValidations NSTextInputClient NSChangeSpelling (NSText) NSIgnoreMisspelledWords (NSText) NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSTextView.h
Companion guides	Text System Overview Text System User Interface Layer Programming Guide for Cocoa
Related sample code	QTSSConnectionMonitor QTSSInspector Quartz Composer WWDC 2005 TextEdit Sketch-112 TextEditPlus

Class at a Glance

`NSTextView` is the front-end component of the Application Kit's text system. It displays and manipulates text laid out in an area defined by an `NSTextContainer` object and adds many features to those defined by its superclass, `NSText`. Many of the methods that you'll use most frequently are declared by the superclass; see the NSText class specification for details.

Instances of this class can be created using Interface Builder or using one of the following methods:

`initWithFrame:` (page 36)
: Creates an `NSTextView` instance along with all of its supporting objects.

`initWithFrame:textContainer:` (page 36)
: Designated initializer.

Commonly Used Methods

The methods most commonly used with `NSTextView` objects are declared in `NSText`, the superclass. These methods provide access to the other major components of the text system:

`textStorage` (page 90)
: Returns the associated NSTextStorage object.

`textContainer` (page 89)
: Returns the associated NSTextContainer object.

`layoutManager` (page 42)
: Returns the associated NSLayoutManager object.

Overview

`NSTextView` is the front-end class to the Application Kit's text system. It draws the text managed by the back-end components and handles user events to select and modify its text. `NSTextView` is the principal means to obtain a text object that caters to almost all needs for displaying and managing text at the user interface level. While `NSTextView` is a subclass of `NSText`—which declares the most general Cocoa interface to the text system—`NSTextView` adds major features beyond the capabilities of `NSText`.

About Delegate Methods

`NSTextView` communicates with its delegate through methods declared both by `NSTextView` and by its superclass, `NSText`. See the NSText class specification for those other delegate methods. Note that all delegation messages come from the first text view.

Adopted Protocols

NSTextInput

- `attributedSubstringFromRange:`
- `characterIndexForPoint:`
- `conversationIdentifier`
- `doCommandBySelector:`
- `firstRectForCharacterRange:`
- `hasMarkedText`
- `insertText:`
- `markedRange`
- `selectedRange`
- `setMarkedText:selectedRange:`
- `unmarkText`
- `validAttributesForMarkedText`

NSUserInterfaceValidations

- validateUserInterfaceItem:

Tasks

Initializing

- initWithFrame:textContainer: (page 36)
  Initializes a text view.
- initWithFrame: (page 36)
  Initializes a text view.

Registering Services Information

+ registerForServices (page 22)
  Registers send and return types for the Services facility.

Accessing Text System Objects

- setTextContainer: (page 79)
  Sets the receiver's text container.
- replaceTextContainer: (page 54)
  Replaces the text container for the group of text system objects containing the receiver, keeping the association between the receiver and its layout manager intact.
- textContainer (page 89)
  Returns the receiver's text container.
- setTextContainerInset: (page 80)
  Sets the empty space the receiver leaves around its associated text container.
- textContainerInset (page 90)
  Returns the empty space the receiver leaves around its text container.
- textContainerOrigin (page 90)
  Returns the origin of the receiver's text container.
- invalidateTextContainerOrigin (page 38)
  Invalidates the calculated origin of the text container.
- layoutManager (page 42)
  Returns the layout manager that lays out text for the receiver's text container.
- textStorage (page 90)
  Returns the receiver's text storage object.
- undoManagerForTextView: (page 114) *delegate method*
  Returns the undo manager for the specified text view.

Setting Graphics Attributes

- `setBackgroundColor:` (page 66)
  Sets the receiver's background color.
- `backgroundColor` (page 26)
  Returns the receiver's background color.
- `setDrawsBackground:` (page 69)
  Sets whether the receiver draws its background.
- `drawsBackground` (page 34)
  Returns whether the receiver draws its background
- `setAllowsDocumentBackgroundColorChange:` (page 63)
  Sets whether the receiver allows its background color to change.
- `allowsDocumentBackgroundColorChange` (page 24)
  Returns whether the receiver allows its background color to change.
- `changeDocumentBackgroundColor:` (page 27)
  An action method used to set the background color.

Controlling Display

- `setNeedsDisplayInRect:avoidAdditionalLayout:` (page 73)
  Marks the receiver as requiring display.
- `shouldDrawInsertionPoint` (page 84)
  Returns whether the receiver should draw its insertion point.
- `drawInsertionPointInRect:color:turnedOn:` (page 34)
  Draws or erases the insertion point.
- `drawViewBackgroundInRect:` (page 35)
  Draws the background of the text view.
- `setConstrainedFrameSize:` (page 67)
  Attempts to set the frame size as if by user action.
- `cleanUpAfterDragOperation` (page 28)
  Releases the drag information still existing after the dragging session has completed.
- `showFindIndicatorForRange:` (page 85)
  Causes a temporary highlighting effect to appear around the visible portion (or portions) of the specified range.
- `textView:willDisplayToolTip:forCharacterAtIndex:` (page 111) *delegate method*
  Returns the actual tooltip to display.

Inserting Text

- `insertText:` (page 38)
  Inserts *`aString`* into the receiver's text at the insertion point if there is one, otherwise replacing the selection.

- `allowedInputSourceLocales` (page 24)
  Returns an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.
- `setAllowedInputSourceLocales:` (page 63)
  Sets an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.

Setting Behavioral Attributes

- `allowsUndo` (page 25)
  Returns whether the receiver allows undo.
- `setAllowsUndo:` (page 64)
  Sets whether undo support is enabled.
- `setEditable:` (page 69)
  Controls whether the text views sharing the receiver's layout manager allow the user to edit text.
- `isEditable` (page 40)
  Returns whether the text views sharing the receiver's layout manager allow the user to edit text.
- `setSelectable:` (page 74)
  Controls whether the text views sharing the receiver's layout manager allow the user to select text.
- `isSelectable` (page 42)
  Returns whether the text views sharing the receiver's layout manager allow the user to select text.
- `setFieldEditor:` (page 70)
  Controls whether the text views sharing the receiver's layout manager behave as field editors.
- `isFieldEditor` (page 40)
  Returns whether the text views sharing the receiver's layout manager behave as field editors.
- `setRichText:` (page 73)
  Controls whether the text views sharing the receiver's layout manager allow the user to apply attributes to specific ranges of text.
- `isRichText` (page 41)
  Returns whether the text views sharing the receiver's layout manager allow the user to apply attributes to specific ranges of text.
- `setImportsGraphics:` (page 71)
  Controls whether the text views sharing the receiver's layout manager allow the user to import files by dragging.
- `importsGraphics` (page 35)
  Returns whether the text views sharing the receiver's layout manager allow the user to import files by dragging.
- `setBaseWritingDirection:range:` (page 66)
  Sets the base writing direction of a range of text.
- `toggleBaseWritingDirection:` (page 92)
  Changes the base writing direction of a paragraph between left-to-right and right-to-left.
- `setDefaultParagraphStyle:` (page 68)
  Sets the receiver's default paragraph style.
- `defaultParagraphStyle` (page 30)
  Returns the receiver's default paragraph style.

- `outline:` (page 46)
  Adds the outline attribute to the selected text attributes if absent; removes the attribute if present.
- `allowsImageEditing` (page 25)
  Indicates whether image attachments should permit editing of their images.
- `setAllowsImageEditing:` (page 64)
  Specifies whether image attachments should permit editing of their images.
- `setAutomaticQuoteSubstitutionEnabled:` (page 65)
  Enables and disables automatic quotation mark substitution.
- `isAutomaticQuoteSubstitutionEnabled` (page 39)
  Indicates whether automatic quotation mark substitution is enabled.
- `toggleAutomaticQuoteSubstitution:` (page 92)
  Changes the state of automatic quotation mark substitution from enabled to disabled and vice versa.
- `setAutomaticLinkDetectionEnabled:` (page 65)
  Enables or disables automatic link detection.
- `isAutomaticLinkDetectionEnabled` (page 39)
  Indicates whether automatic link detection is enabled.
- `toggleAutomaticLinkDetection:` (page 91)
  Changes the state of automatic link detection from enabled to disabled and vice versa.
- `displaysLinkToolTips` (page 31)
  Indicates whether the text view automatically supplies the destination of a link as a tooltip for text that has a link attribute.
- `setDisplaysLinkToolTips:` (page 68)
  Enables or disables automatic display of link tooltips.

Using the Ruler

- `setUsesRuler:` (page 82)
  Controls whether the text views sharing the receiver's layout manager use a ruler.
- `usesRuler` (page 98)
  Returns whether the text views sharing the receiver's layout manager use a ruler.
- `setRulerVisible:` (page 74)
  Controls whether the scroll view enclosing text views sharing the receiver's layout manager displays the ruler.
- `isRulerVisible` (page 42)
  Returns whether the scroll view enclosing the text views sharing the receiver's layout manager shows its ruler.

Managing the Selection

- `selectedRanges` (page 60)
  Returns an array containing the ranges of characters selected in the receiver's layout manager.
- `setSelectedRange:` (page 75)
  Sets the selection to the characters in a single range.

- `setSelectedRanges:` (page 76)
  Sets the selection to the characters in an array of ranges.
- `setSelectedRange:affinity:stillSelecting:` (page 75)
  Sets the selection to a range of characters in response to user action.
- `setSelectedRanges:affinity:stillSelecting:` (page 77)
  Sets the selection to the characters in an array of ranges in response to user action.
- `selectionAffinity` (page 61)
  Returns the preferred direction of selection.
- `setSelectionGranularity:` (page 78)
  Sets the selection granularity for subsequent extension of a selection.
- `selectionGranularity` (page 61)
  Returns the current selection granularity, used during mouse tracking to modify the range of the selection.
- `setInsertionPointColor:` (page 71)
  Sets the color of the insertion point
- `insertionPointColor` (page 37)
  Returns the color used to draw the insertion point.
- `updateInsertionPointStateAndRestartTimer:` (page 96)
  Updates the insertion point's location and optionally restarts the blinking cursor timer.
- `setSelectedTextAttributes:` (page 77)
  Sets the attributes used to indicate the selection.
- `selectedTextAttributes` (page 60)
  Returns the attributes used to indicate the selection.
- `setMarkedTextAttributes:` (page 72)
  Sets the attributes used to draw marked text.
- `markedTextAttributes` (page 44)
  Returns the attributes used to draw marked text.
- `setLinkTextAttributes:` (page 72)
  Sets the attributes used to draw the onscreen presentation of link text.
- `linkTextAttributes` (page 43)
  Returns the attributes used to draw the onscreen presentation of link text.
- `characterIndexForInsertionAtPoint:` (page 27)
  Returns a character index appropriate for placing a zero-length selection for an insertion point associated with the mouse at the given point.
- `textView:willChangeSelectionFromCharacterRange:toCharacterRange:` (page 110) *delegate method*
  Returns the actual range to select.
- `textView:willChangeSelectionFromCharacterRanges:toCharacterRanges:` (page 110) *delegate method*
  Returns the actual character ranges to select.
- `textViewDidChangeSelection:` (page 113) *delegate method*
  Sent when the selection changes in the text view.

Managing the Pasteboard

- `preferredPasteboardTypeFromArray:restrictedToTypesFromArray:` (page 48)
  Returns whatever type on the pasteboard would be most preferred for copying data.
- `readSelectionFromPasteboard:` (page 53)
  Reads the text view's preferred type of data from the specified pasteboard.
- `readSelectionFromPasteboard:type:` (page 54)
  Reads data of the given type from the specified pasteboard.
- `readablePasteboardTypes` (page 52)
  Returns the types this text view can read immediately from the pasteboard.
- `writablePasteboardTypes` (page 100)
  Returns the pasteboard types that can be provided from the current selection.
- `writeSelectionToPasteboard:type:` (page 100)
  Writes the current selection to the specified pasteboard using the given type.
- `writeSelectionToPasteboard:types:` (page 101)
  Writes the current selection to the specified pasteboard under each given type.
- `validRequestorForSendType:returnType:` (page 99)
  Returns `self` if the text view can provide and accept the specified data types, or `nil` if it can't.
- `textView:writablePasteboardTypesForCell:atIndex:` (page 112) *delegate method*
  Returns the writable pasteboard types for a given cell.
- `textView:writeCell:atIndex:toPasteboard:type:` (page 112) *delegate method*
  Returns whether data of the specified type for the given cell could be written to the specified pasteboard.

Setting Text Attributes

- `alignJustified:` (page 24)
  Applies full justification to selected paragraphs (or all text, if the receiver is a plain text object).
- `changeAttributes:` (page 26)
  Changes the attributes of the current selection.
- `changeColor:` (page 27)
  Sets the color of the selected text.
- `setAlignment:range:` (page 62)
  Sets the alignment of the paragraphs containing characters in the specified range.
- `setTypingAttributes:` (page 81)
  Sets the receiver's typing attributes.
- `typingAttributes` (page 95)
  Returns the current typing attributes.
- `useStandardKerning:` (page 99)
  Set the receiver to use pair kerning data for the glyphs in its selection, or for all glyphs if the receiver is a plain text view.
- `lowerBaseline:` (page 44)
  Lowers the baseline offset of selected text by 1 point, or of all text if the receiver is a plain text view.

- `raiseBaseline:` (page 48)
  Raises the baseline offset of selected text by 1 point, or of all text if the receiver is a plain text view.
- `turnOffKerning:` (page 94)
  Sets the receiver to use nominal glyph spacing for the glyphs in its selection, or for all glyphs if the receiver is a plain text view.
- `loosenKerning:` (page 43)
  Increases the space between glyphs in the receiver's selection, or in all text if the receiver is a plain text view.
- `tightenKerning:` (page 91)
  Decreases the space between glyphs in the receiver's selection, or for all glyphs if the receiver is a plain text view.
- `useStandardLigatures:` (page 99)
  Sets the receiver to use the standard ligatures available for the fonts and languages used when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.
- `turnOffLigatures:` (page 95)
  Sets the receiver to use only required ligatures when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.
- `useAllLigatures:` (page 97)
  Sets the receiver to use all ligatures available for the fonts and languages used when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.
- `toggleTraditionalCharacterShape:` (page 94)
  Toggles the `NSCharacterShapeAttributeName` attribute at the current selection.
- `textView:shouldChangeTextInRange:replacementString:` (page 107) *delegate method*
  Sent when a text view needs to determine if text in a specified range should be changed.
- `textView:shouldChangeTextInRanges:replacementStrings:` (page 108) *delegate method*
  Sent when a text view needs to determine if text in an array of specified ranges should be changed.
- `textView:shouldChangeTypingAttributes:toAttributes:` (page 109) *delegate method*
  Sent when the typing attributes are changed.
- `textViewDidChangeTypingAttributes:` (page 113) *delegate method*
  Sent when a text view's typing attributes change.

Clicking and Pasting

- `clickedOnLink:atIndex:` (page 28)
  Causes the text view to act as if the user clicked on some text with the given link as the value of a link attribute associated with the text.
- `pasteAsPlainText:` (page 46)
  Inserts the contents of the pasteboard into the receiver's text as plain text.
- `pasteAsRichText:` (page 47)
  This action method inserts the contents of the pasteboard into the receiver's text as rich text, maintaining its attributes.
- `textView:clickedOnCell:inRect:atIndex:` (page 102) *delegate method*
  Sent when the user clicks a cell.

- `textView:doubleClickedOnCell:inRect:atIndex:` (page 106) *delegate method*
  Sent when the user double-clicks a cell.
- `textView:clickedOnLink:atIndex:` (page 103) *delegate method*
  Sent after the user clicks a link.

Undo Support

- `breakUndoCoalescing` (page 26)
  Informs the receiver that it should begin coalescing successive typing operations in a new undo grouping.

Methods for Subclasses to Use or Override

- `updateFontPanel` (page 96)
  Updates the Font panel to contain the font attributes of the selection.
- `updateRuler` (page 97)
  Updates the ruler view in the receiver's enclosing scroll view to reflect the selection's paragraph and marker attributes.
- `acceptableDragTypes` (page 23)
  Returns the data types that the receiver accepts as the destination view of a dragging operation.
- `updateDragTypeRegistration` (page 95)
  Updates the acceptable drag types of all text views associated with the receiver's layout manager.
- `selectionRangeForProposedRange:granularity:` (page 61)
  Returns an adjusted selected range based on the selection granularity.
- `rangeForUserCharacterAttributeChange` (page 49)
  Returns the range of characters affected by an action method that changes character (not paragraph) attributes.
- `rangesForUserCharacterAttributeChange` (page 51)
  Returns an array containing the ranges of characters affected by an action method that changes character (not paragraph) attributes.
- `rangeForUserParagraphAttributeChange` (page 50)
  Returns the range of characters affected by an action method that changes paragraph (not character) attributes.
- `rangesForUserParagraphAttributeChange` (page 52)
  Returns an array containing the ranges of characters affected by a method that changes paragraph (not character) attributes.
- `rangeForUserTextChange` (page 51)
  Returns the range of characters affected by a method that changes characters (as opposed to attributes).
- `rangesForUserTextChange` (page 52)
  Returns an array containing the ranges of characters affected by a method that changes characters (as opposed to attributes).
- `shouldChangeTextInRange:replacementString:` (page 83)
  Initiates a series of delegate messages (and general notifications) to determine whether modifications can be made to the characters and attributes of the receiver's text.

- `shouldChangeTextInRanges:replacementStrings:` (page 84)
  Initiates a series of delegate messages (and general notifications) to determine whether modifications can be made to the characters and attributes of the receiver's text.
- `didChangeText` (page 31)
  Sends out necessary notifications when a text change completes.
- `setSmartInsertDeleteEnabled:` (page 78)
  Controls whether the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation.
- `smartInsertDeleteEnabled` (page 87)
  Returns whether the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation.
- `smartDeleteRangeForProposedRange:` (page 85)
  Returns an extended range that includes adjacent whitespace that should be deleted along with the proposed range in order to preserve proper spacing and punctuation.
- `smartInsertAfterStringForString:replacingRange:` (page 86)
  Returns any whitespace that needs to be added after the string to preserve proper spacing and punctuation when the string replaces the characters in the specified range.
- `smartInsertBeforeStringForString:replacingRange:` (page 86)
  Returns any whitespace that needs to be added before the string to preserve proper spacing and punctuation when the string replaces the characters in the specified range.
- `smartInsertForString:replacingRange:beforeString:afterString:` (page 87)
  Determines whether whitespace needs to be added around the string to preserve proper spacing and punctuation when it replaces the characters in the specified range.
- `toggleSmartInsertDelete:` (page 93)
  Changes the state of smart insert and delete from enabled to disabled and vice versa.

Working With the Spelling Checker

- `isContinuousSpellCheckingEnabled` (page 40)
  Indicates whether the receiver has continuous spell checking enabled.
- `setContinuousSpellCheckingEnabled:` (page 67)
  Enables or disables continuous spell checking.
- `spellCheckerDocumentTag` (page 88)
  Returns a tag identifying the text view's text as a document for the spell checker server.
- `toggleContinuousSpellChecking:` (page 93)
  Toggles whether continuous spell checking is enabled for the receiver.
- `setGrammarCheckingEnabled:` (page 70)
  Enables and disables grammar checking.
- `isGrammarCheckingEnabled` (page 41)
  Indicates whether or not grammar checking is enabled.
- `toggleGrammarChecking:` (page 93)
  Changes the state of grammar checking from enabled to disabled and vice versa.
- `setSpellingState:range:` (page 79)
  Sets the spelling state, which controls the display of the spelling and grammar indicators on the given text range.

- `textView:shouldSetSpellingState:range:` (page 109) *delegate method*
  Sent when the spelling state is changed.

NSRulerView Client Methods

- `rulerView:didMoveMarker:` (page 55)
  Modifies the paragraph style of the paragraphs containing the selection to record the new location of the marker.
- `rulerView:willMoveMarker:toLocation:` (page 59)
  Returns a potentially modified location to which the marker should be moved.
- `rulerView:shouldMoveMarker:` (page 57)
  Returns whether the marker should be moved.
- `rulerView:didRemoveMarker:` (page 56)
  Modifies the paragraph style of the paragraphs containing the selection—if possible—by removing the specified marker.
- `rulerView:shouldRemoveMarker:` (page 58)
  Returns whether the marker should be removed.
- `rulerView:didAddMarker:` (page 55)
  Modifies the paragraph style of the paragraphs containing the selection to accommodate a new marker.
- `rulerView:shouldAddMarker:` (page 57)
  Returns whether a new marker can be added.
- `rulerView:willAddMarker:atLocation:` (page 59)
  Returns a potentially modified location to which the marker should be added.
- `rulerView:handleMouseDown:` (page 56)
  Adds a left tab marker to the ruler at the location clicked.

Assigning a Delegate

- `setDelegate:` (page 68)
  Sets the delegate for all text views sharing the receiver's layout manager.
- `delegate` (page 30)
  Returns the delegate used by the receiver and all other text views sharing the receiver's layout manager.

Dragging

- `dragImageForSelectionWithEvent:origin:` (page 32)
  Returns an appropriate drag image for the drag initiated by the specified event.
- `dragOperationForDraggingInfo:type:` (page 32)
  Returns the type of drag operation that should be performed if the image were released now.
- `dragSelectionWithEvent:offset:slideBack:` (page 33)
  Begins dragging the current selected text range.

- `acceptsGlyphInfo` (page 23)
  Returns whether the receiver accepts the glyph info attribute.
- `setAcceptsGlyphInfo:` (page 62)
  Sets whether the receiver accepts the glyph info attribute.
- `textView:draggedCell:inRect:event:atIndex:` (page 107) *delegate method*
  Sent when the user attempts to drag a cell.

Speaking Text

- `startSpeaking:` (page 88)
  Speaks the selected text, or all text if no selection.
- `stopSpeaking:` (page 89)
  Stops the speaking of text.

Working with Panels

- `setUsesFontPanel:` (page 82)
  Controls whether the text views sharing the receiver's layout manager use the Font panel and Font menu.
- `usesFontPanel` (page 98)
  Returns whether the text views sharing the receiver's layout manager use the Font panel.
- `setUsesFindPanel:` (page 81)
  Specifies whether the receiver allows for a find panel.
- `usesFindPanel` (page 97)
  Returns whether the receiver allows for a find panel.
- `performFindPanelAction:` (page 47)
  Performs a find panel action specified by the sender's tag.
- `orderFrontLinkPanel:` (page 45)
  Brings forward a panel allowing the user to manipulate links in the text view.
- `orderFrontListPanel:` (page 45)
  Brings forward a panel allowing the user to manipulate text lists in the text view.
- `orderFrontSpacingPanel:` (page 45)
  Brings forward a panel allowing the user to manipulate text line heights, interline spacing, and paragraph spacing, in the text view.
- `orderFrontTablePanel:` (page 46)
  Brings forward a panel allowing the user to manipulate text tables in the text view.

Text Completion

- `complete:` (page 29)
  Invokes completion in a text view.

- `completionsForPartialWordRange:indexOfSelectedItem:` (page 29)
  Returns an array of potential completions, in the order to be presented, representing possible word completions available from a partial word.
- `insertCompletion:forPartialWordRange:movement:isFinal:` (page 37)
  Inserts the selected completion into the text at the appropriate location.
- `rangeForUserCompletion` (page 49)
  Returns the partial range from the most recent beginning of a word up to the insertion point.
- `textView:completions:forPartialWordRange:indexOfSelectedItem:` (page 104) *delegate method*
  Returns the actual completions for a partial word.

Performing Commands

- `textView:doCommandBySelector:` (page 105) *delegate method*
  Sent to allow the delegate to perform the command for the text view.

Deprecated Methods

- `textView:clickedOnLink:` (page 103) *delegate method*
  Sent after the user clicks on a link. (Deprecated. Use `textView:clickedOnLink:atIndex:` (page 103) instead.)
- `textView:draggedCell:inRect:event:` (page 106) *delegate method*
  Sent when the user attempts to drag a cell. (Deprecated. Use `textView:draggedCell:inRect:event:atIndex:` (page 107) instead.)
- `textView:clickedOnCell:inRect:` (page 102) *delegate method*
  Sent when the user clicks a cell. (Deprecated. Use `textView:clickedOnCell:inRect:atIndex:` (page 102) instead.)
- `textView:doubleClickedOnCell:inRect:` (page 105) *delegate method*
  Sent when the user double-clicks a cell. (Deprecated. Use `textView:doubleClickedOnCell:inRect:atIndex:` (page 106) instead.)

Class Methods

registerForServices

Registers send and return types for the Services facility.

```
+ (void)registerForServices
```

Discussion
This method is invoked automatically when the first instance of a text view is created; you should never need to invoke it directly.

Subclasses of `NSTextView` that wish to add support for new service types should override `registerForServices` to call `super` and then register their own new types.

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

Declared In
`NSTextView.h`

Instance Methods

acceptableDragTypes

Returns the data types that the receiver accepts as the destination view of a dragging operation.

```
- (NSArray *)acceptableDragTypes
```

Return Value
The data types that the receiver accepts as the destination view of a dragging operation.

Discussion
These types are automatically registered as necessary by the text view. Subclasses should override this method as necessary to add their own types to those returned by `NSTextView`'s implementation. They must then also override the appropriate methods of the `NSDraggingDestination` protocol to support import of those types. See that protocol's specification for more information.

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

See Also
`- updateDragTypeRegistration` (page 95)

Declared In
`NSTextView.h`

acceptsGlyphInfo

Returns whether the receiver accepts the glyph info attribute.

```
- (BOOL)acceptsGlyphInfo
```

Return Value
`YES` if the receiver accepts the `NSGlyphInfoAttributeName` attribute from text input sources such as input methods and the pasteboard, `NO` otherwise.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- setAcceptsGlyphInfo:` (page 62)

Declared In
`NSTextView.h`

alignJustified:

Applies full justification to selected paragraphs (or all text, if the receiver is a plain text object).

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

Parameters

sender

The control that sent the message; may be `nil`.

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

See Also
- `alignCenter:` (NSText)
- `alignLeft:` (NSText)
- `alignRight:` (NSText)
- `alignment` (NSText)
- `setAlignment:` (NSText)

Declared In
`NSTextView.h`

allowedInputSourceLocales

Returns an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.

```
- (NSArray *)allowedInputSourceLocales
```

Return Value
The locale identifiers of allowed input sources.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setAllowedInputSourceLocales:` (page 63)

Declared In
`NSTextView.h`

allowsDocumentBackgroundColorChange

Returns whether the receiver allows its background color to change.

```
- (BOOL)allowsDocumentBackgroundColorChange
```

Return Value
`YES` if the receiver allows its background color to change, otherwise `NO`.

Discussion
This corresponds to the background color of the entirety of the text view, not just to a selected range of text.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setAllowsDocumentBackgroundColorChange:` (page 63)
- `changeDocumentBackgroundColor:` (page 27)

Declared In
`NSTextView.h`

allowsImageEditing

Indicates whether image attachments should permit editing of their images.

```
- (BOOL)allowsImageEditing
```

Return Value
`YES` if image editing is allowed; otherwise, `NO`.

Discussion
For image editing to be allowed, the text view must be editable and the text attachment cell must support image editing.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setAllowsImageEditing:` (page 64)

Declared In
`NSTextView.h`

allowsUndo

Returns whether the receiver allows undo.

```
- (BOOL)allowsUndo
```

Return Value
`YES` if the receiver allows undo, otherwise `NO`.

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

See Also
- `setAllowsUndo:` (page 64)

Declared In
`NSTextView.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 34)
- `setBackgroundColor:` (page 66)

Declared In
`NSTextView.h`

breakUndoCoalescing

Informs the receiver that it should begin coalescing successive typing operations in a new undo grouping.

```
- (void)breakUndoCoalescing
```

Special Considerations

This method should be invoked when saving the receiver's contents to preserve proper tracking of unsaved changes and the document's dirty state.

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

Declared In
`NSTextView.h`

changeAttributes:

Changes the attributes of the current selection.

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

Parameters

sender

The control that sent the message. Must respond to `convertAttributes:`.

Discussion
This method changes the attributes by invoking `convertAttributes:` on *sender* and applying the returned attributes to the appropriate text. See the `NSFontManager` class reference for more information on attribute conversion.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTextView.h

changeColor:

Sets the color of the selected text.

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

Parameters

sender

The control that sent the message. NSTextView's implementation sends a color message to *sender* to get the new color.

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

Declared In
NSTextView.h

changeDocumentBackgroundColor:

An action method used to set the background color.

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

Parameters

sender

The control that wants to set the background color.

Discussion
This method gets the new color by sending a color message to *sender*.

This will only set the background color if allowsDocumentBackgroundColorChange (page 24)returns YES.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setAllowsDocumentBackgroundColorChange: (page 63)
- allowsDocumentBackgroundColorChange (page 24)

Declared In
NSTextView.h

characterIndexForInsertionAtPoint:

Returns a character index appropriate for placing a zero-length selection for an insertion point associated with the mouse at the given point.

```
- (NSUInteger)characterIndexForInsertionAtPoint:(NSPoint)point
```

Parameters

point

The point for which to return an index, in view coordinates.

Return Value

The character index for the insertion point.

Discussion

This method should be used for insertion points associated with mouse clicks, drag events, and so forth. For other purposes, it is better to use `NSLayoutManager` methods.

The `NSTextInput` method `characterIndexForPoint:` is not suitable for this role; it is intended only for uses related to text input methods.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSTextView.h`

cleanUpAfterDragOperation

Releases the drag information still existing after the dragging session has completed.

```
- (void)cleanUpAfterDragOperation
```

Discussion

Subclasses may override this method to clean up any additional data structures used for dragging. In your overridden method, be sure to invoke `super`'s implementation of this method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextView.h`

clickedOnLink:atIndex:

Causes the text view to act as if the user clicked on some text with the given link as the value of a link attribute associated with the text.

```
- (void)clickedOnLink:(id)link atIndex:(NSUInteger)charIndex
```

Parameters

link

The link that was clicked; the value of `NSLinkAttributeName`.

charIndex

The character index where the click occurred, indexed within the text storage.

Discussion

If, for instance, you have a special attachment cell that can follow links, you can use this method to ask the text view to follow a link once you decide it should. In addition, this method is invoked by the text view during mouse tracking if the user clicks a link.

The *charIndex* parameter is a character index somewhere in the range of the link attribute. If the user actually physically clicked the link, then it should be the character that was originally clicked. In some cases a link may be opened indirectly or programmatically, in which case a character index somewhere in the range of the link attribute is supplied.

This method sends the `textView:clickedOnLink:atIndex:` (page 103) delegate message if the delegate implements it, so that the delegate can handle the click.

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

See Also
`- textView:clickedOnLink:atIndex:` (page 103) (delegate method)

Declared In
`NSTextView.h`

complete:

Invokes completion in a text view.

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

Parameters
sender
The control sending the message. May be `nil`.

Discussion
By default invoked using the F5 key, this method provides users with a choice of completions for the word currently being typed. May be invoked programmatically if autocompletion is desired by a client of the text system. You can change the key invoking this method using the text system's key bindings mechanism; see "Text System Defaults and Key Bindings" for an explanation of the procedure.

The delegate may replace or modify the list of possible completions by implementing `textView:completions:forPartialWordRange:indexOfSelectedItem:` (page 104). Subclasses may control the list by overriding `completionsForPartialWordRange:indexOfSelectedItem:` (page 29).

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
SearchField

Declared In
`NSTextView.h`

completionsForPartialWordRange:indexOfSelectedItem:

Returns an array of potential completions, in the order to be presented, representing possible word completions available from a partial word.

```
- (NSArray *)completionsForPartialWordRange:(NSRange)charRange
    indexOfSelectedItem:(NSInteger *)index
```

Parameters

charRange

The range of characters of the partial word to be completed.

index

On return, optionally set to the completion that should be initially selected. The default is 0, and –1 indicates no selection.

Return Value

An array of potential completions, in the order to be presented, representing possible word completions available from a partial word at *charRange*. Returning `nil` or a zero-length array suppresses completion.

Discussion

May be overridden by subclasses to modify or override the list of possible completions.

This method should call the delegate method `textView:completions:forPartialWordRange:indexOfSelectedItem:` (page 104) if the delegate implements such a method.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`NSTextView.h`

defaultParagraphStyle

Returns the receiver's default paragraph style.

```
- (NSParagraphStyle *)defaultParagraphStyle
```

Return Value

The receiver's default paragraph style.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- setDefaultParagraphStyle:` (page 68)

Declared In

`NSTextView.h`

delegate

Returns the delegate used by the receiver and all other text views sharing the receiver's layout manager.

```
- (id)delegate
```

Return Value

The delegate used by the receiver and all other text views sharing the receiver's layout manager, or `nil` if there is none.

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

See Also
- `setDelegate:` (page 68)

Related Sample Code
TextLinks

Declared In
`NSTextView.h`

didChangeText

Sends out necessary notifications when a text change completes.

```
- (void)didChangeText
```

Discussion
Invoked automatically at the end of a series of changes, this method posts an `NSTextDidChangeNotification` to the default notification center, which also results in the delegate receiving an `NSText` delegate `textDidChange:` message.

Subclasses implementing methods that change their text should invoke this method at the end of those methods. See Subclassing NSTextView for more information.

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

See Also
- `shouldChangeTextInRange:replacementString:` (page 83)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
`NSTextView.h`

displaysLinkToolTips

Indicates whether the text view automatically supplies the destination of a link as a tooltip for text that has a link attribute.

```
- (BOOL)displaysLinkToolTips
```

Return Value
`YES` if link tooltips are automatically displayed; otherwise, `NO`.

Discussion
The default value for this feature is `YES`; clients who do not wish tooltips to be displayed automatically must explicitly disable it.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setDisplaysLinkToolTips:` (page 68)

Declared In
`NSTextView.h`

dragImageForSelectionWithEvent:origin:

Returns an appropriate drag image for the drag initiated by the specified event.

```
- (NSImage *)dragImageForSelectionWithEvent:(NSEvent *)event
    origin:(NSPointPointer)origin
```

Parameters

event
The event that initiated the drag session.

origin
On return, the lower-left point of the image in view coordinates.

Return Value
An appropriate drag image for the drag initiated by *event*. May be `nil`, in which case a default icon will be used.

Discussion
This method is used by `dragSelectionWithEvent:offset:slideBack:` (page 33). It can be called by others who need such an image, or can be overridden by subclasses to return a different image.

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

Declared In
`NSTextView.h`

dragOperationForDraggingInfo:type:

Returns the type of drag operation that should be performed if the image were released now.

```
- (NSDragOperation)dragOperationForDraggingInfo:(id < NSDraggingInfo >)dragInfo
    type:(NSString *)type
```

Parameters

dragInfo
The drag information.

type
The pasteboard type that will be read from the dragging pasteboard.

Return Value
The drag operation that should be performed if the image were released now.

Discussion
The returned value should be one of the following:

Option	Meaning
`NSDragOperationCopy`	The data represented by the image will be copied.
`NSDragOperationLink`	The data will be shared.
`NSDragOperationGeneric`	The operation will be defined by the destination.
`NSDragOperationPrivate`	The operation is negotiated privately between the source and the destination.

If none of the operations is appropriate, this method should return `NSDragOperationNone`.

This method is called repeatedly from `draggingEntered:` and `draggingUpdated:` as the user drags the image.

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

See Also
- `draggingEntered:` (NSDraggingDestination)
- `draggingUpdated:` (NSDraggingDestination)

Declared In
`NSTextView.h`

dragSelectionWithEvent:offset:slideBack:

Begins dragging the current selected text range.

```
- (BOOL)dragSelectionWithEvent:(NSEvent *)event offset:(NSSize)mouseOffset
    slideBack:(BOOL)slideBack
```

Parameters

event
The event that initiated dragging the selection.

mouseOffset
The cursor's current location relative to the mouse-down *event*.

slideBack
`YES` if the image being dragged should slide back to its original position if the drag does not succeed, `NO` otherwise.

Return Value
`YES` if the drag can be successfully initiated, `NO` otherwise.

Discussion
Primarily for subclasses, who can override it to intervene at the beginning of a drag.

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

Declared In
NSTextView.h

drawInsertionPointInRect:color:turnedOn:

Draws or erases the insertion point.

```
- (void)drawInsertionPointInRect:(NSRect)aRect color:(NSColor *)aColor
    turnedOn:(BOOL)flag
```

Parameters

aRect
The rectangle in which to draw the insertion point.

aColor
The color with which to draw the insertion point.

flag
YES to draw the insertion point, NO to erase it.

Special Considerations

The focus must be locked on the receiver when this method is invoked. You should not need to invoke this method directly.

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

See Also
- insertionPointColor (page 37)
- shouldDrawInsertionPoint (page 84)
- backgroundColor (page 26)
- lockFocus (NSView)

Declared In
NSTextView.h

drawsBackground

Returns whether the receiver draws its background

```
- (BOOL)drawsBackground
```

Return Value
YES if the receiver draws its background, NO otherwise.

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

See Also
- backgroundColor (page 26)
- setDrawsBackground: (page 69)

Declared In
NSTextView.h

drawViewBackgroundInRect:

Draws the background of the text view.

```
- (void)drawViewBackgroundInRect:(NSRect)rect
```

Parameters

rect

The rectangle in which to draw the background.

Discussion
Subclasses can override this method to perform additional drawing behind the text.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTextView.h

importsGraphics

Returns whether the text views sharing the receiver's layout manager allow the user to import files by dragging.

```
- (BOOL)importsGraphics
```

Return Value
YES if the user is allowed to import files by dragging onto the text views sharing the receiver's layout manager, NO otherwise.

Discussion
Text views that are set to accept dragged files are also set to allow rich text. By default, text views don't accept dragged files but do allow rich text.

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

See Also
- isRichText (page 41)
- textStorage (page 90)
+ attributedStringWithAttachment: (NSAttributedString Additions)
- insertAttributedString:atIndex: (NSMutableAttributedString)
- setImportsGraphics: (page 71)

Declared In
NSTextView.h

initWithFrame:

Initializes a text view.

```
- (id)initWithFrame:(NSRect)frameRect
```

Parameters

frameRect

The frame rectangle of the text view.

Return Value

An initialized text view.

Discussion

This method creates the entire collection of objects associated with a text view—its text container, layout manager, and text storage—and invokes `initWithFrame:textContainer:` (page 36).

This method creates the text web in such a manner that the text view is the principal owner of the objects in the web.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextView.h`

initWithFrame:textContainer:

Initializes a text view.

```
- (id)initWithFrame:(NSRect)frameRect textContainer:(NSTextContainer *)aTextContainer
```

Parameters

frameRect

The frame rectangle of the text view.

aTextContainer

The text container of the text view.

Return Value

An initialized text view.

Discussion

This method is the designated initializer for `NSTextView` objects.

Unlike `initWithFrame:` (page 36), which builds up an entire group of text-handling objects, you use this method after you've created the other components of the text-handling system—a text storage object, a layout manager, and a text container. Assembling the components in this fashion means that the text storage, not the text view, is the principal owner of the component objects.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- initWithFrame:` (page 36)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
Sketch-112
TextEditPlus
TextLayoutDemo
TextViewConfig

Declared In
NSTextView.h

insertCompletion:forPartialWordRange:movement:isFinal:

Inserts the selected completion into the text at the appropriate location.

```
- (void)insertCompletion:(NSString *)word forPartialWordRange:(NSRange)charRange
    movement:(NSInteger)movement isFinal:(BOOL)flag
```

Parameters

word
The completion to insert.

charRange
The character range of the text being completed.

movement
The direction of movement. For possible values see the NSText Constants section. This value allows subclasses to distinguish between canceling completion and selection by arrow keys, by return, by tab, or by other means such as clicking.

flag
NO while the user navigates through the potential text completions, YES when a completion is definitively selected or cancelled and the original value is reinserted.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTextView.h

insertionPointColor

Returns the color used to draw the insertion point.

```
- (NSColor *)insertionPointColor
```

Return Value
The color used to draw the insertion point.

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

See Also
- drawInsertionPointInRect:color:turnedOn: (page 34)
- shouldDrawInsertionPoint (page 84)

- setInsertionPointColor: (page 71)

Declared In
NSTextView.h

insertText:

Inserts *aString* into the receiver's text at the insertion point if there is one, otherwise replacing the selection.

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

Parameters

aString
The string to insert. *aString* can be either an NSString object or an NSAttributedString object.

Discussion
The inserted text is assigned the current typing attributes.

This method is the means by which text typed by the user enters an NSTextView. See the NSInputManager class and NSTextInput protocol specifications for more information.

This method is the entry point for inserting text typed by the user and is generally not suitable for other purposes. Programmatic modification of the text is best done by operating on the text storage directly. Because this method pertains to the actions of the user, the text view must be editable for the insertion to work.

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

See Also
- typingAttributes (page 95)

Related Sample Code
SampleScannerApp
TextViewDelegate
WebKitPluginWithSimpleGUI

Declared In
NSTextView.h

invalidateTextContainerOrigin

Invalidates the calculated origin of the text container.

```
- (void)invalidateTextContainerOrigin
```

Discussion
This method is invoked automatically; you should never need to invoke it directly. Usually called because the text view has been resized or the contents of the text container have changed.

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

See Also
- `textContainer` (page 89)
- `textContainerOrigin` (page 90)

Declared In
`NSTextView.h`

isAutomaticLinkDetectionEnabled

Indicates whether automatic link detection is enabled.

```
- (BOOL)isAutomaticLinkDetectionEnabled
```

Return Value
`YES` if automatic link detection is enabled; otherwise, `NO`.

Discussion
Automatic link detection causes strings representing URLs typed in the view to be automatically made into links to those URLs.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setAutomaticLinkDetectionEnabled:` (page 65)
- `toggleAutomaticLinkDetection:` (page 91)

`URLAtIndex:effectiveRange:` (`NSAttributedString`)

Declared In
`NSTextView.h`

isAutomaticQuoteSubstitutionEnabled

Indicates whether automatic quotation mark substitution is enabled.

```
- (BOOL)isAutomaticQuoteSubstitutionEnabled
```

Return Value
`YES` if automatic quotation mark substitution is enabled; otherwise, `NO`.

Discussion
Automatic quote substitution causes ASCII quotation marks and apostrophes to be automatically replaced, on a context-dependent basis, with more typographically accurate symbols.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setAutomaticQuoteSubstitutionEnabled:` (page 65)
- `toggleAutomaticQuoteSubstitution:` (page 92)

Declared In
`NSTextView.h`

isContinuousSpellCheckingEnabled

Indicates whether the receiver has continuous spell checking enabled.

```
- (BOOL)isContinuousSpellCheckingEnabled
```

Return Value
`YES` if the receiver has continuous spell checking enabled, otherwise,`NO`.

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

See Also
- `setContinuousSpellCheckingEnabled:` (page 67)
- `toggleContinuousSpellChecking:` (page 93)

Declared In
`NSTextView.h`

isEditable

Returns whether the text views sharing the receiver's layout manager allow the user to edit text.

```
- (BOOL)isEditable
```

Return Value
`YES` if the text views sharing the receiver's layout manager allow the user to edit text, `NO` otherwise.

Discussion
If a text view is editable, it's also selectable.

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

See Also
- `isSelectable` (page 42)
- `setEditable:` (page 69)

Declared In
`NSTextView.h`

isFieldEditor

Returns whether the text views sharing the receiver's layout manager behave as field editors.

```
- (BOOL)isFieldEditor
```

Return Value
`YES` if the text views sharing the receiver's layout manager behave as field editors, `NO` otherwise.

Discussion
Field editors interpret Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder. Non-field editors instead accept these characters as text input. See Text Fields, Text Views, and the Field Editor for more information on field editors. By default, text views don't behave as field editors.

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

See Also
- `setFieldEditor:` (page 70)

Declared In
`NSTextView.h`

isGrammarCheckingEnabled

Indicates whether or not grammar checking is enabled.

```
- (BOOL)isGrammarCheckingEnabled
```

Return Value
`YES` if grammar checking is enabled; otherwise, `NO`.

Discussion
If grammar checking is enabled, then it is performed alongside spell checking, whenever the text view checks spelling, whether continuously or manually.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setGrammarCheckingEnabled:` (page 70)
- `toggleGrammarChecking:` (page 93)

Declared In
`NSTextView.h`

isRichText

Returns whether the text views sharing the receiver's layout manager allow the user to apply attributes to specific ranges of text.

```
- (BOOL)isRichText
```

Return Value
`YES` if the user is allowed to apply attributes to specific ranges of text in text views sharing the receiver's layout manager, `NO` otherwise.

Discussion
Text fields that don't allow rich text also don't accept dragged files. By default, text views let the user apply multiple attributes to text, but don't accept dragged files.

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

See Also
- `importsGraphics` (page 35)
- `textStorage` (page 90)

- setRichText: (page 73)

Declared In
NSTextView.h

isRulerVisible

Returns whether the scroll view enclosing the text views sharing the receiver's layout manager shows its ruler.

```
- (BOOL)isRulerVisible
```

Return Value
YES if the scroll view enclosing the text views sharing the receiver's layout manager shows its ruler, NO otherwise. The default is NO.

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

See Also
- usesRuler (page 98)
- setRulerVisible: (page 74)
- toggleRuler: (NSText)

Declared In
NSTextView.h

isSelectable

Returns whether the text views sharing the receiver's layout manager allow the user to select text.

```
- (BOOL)isSelectable
```

Return Value
YES if the user is allowed to select text of all text views sharing the receiver's layout manager, NO otherwise.

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

See Also
- isEditable (page 40)
- setSelectable: (page 74)

Declared In
NSTextView.h

layoutManager

Returns the layout manager that lays out text for the receiver's text container.

```
- (NSLayoutManager *)layoutManager
```

Return Value
The layout manager that lays out text for the receiver's text container, or `nil` if there's no such object, such as when a text view isn't linked into a group of text objects.

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

See Also
- `textContainer` (page 89)
- `setLayoutManager:` (NSTextContainer)
- `replaceLayoutManager:` (NSTextContainer)

Related Sample Code
LayoutManagerDemo
Sketch-112

Declared In
`NSTextView.h`

linkTextAttributes

Returns the attributes used to draw the onscreen presentation of link text.

```
- (NSDictionary *)linkTextAttributes
```

Return Value
A dictionary of attributes corresponding to the onscreen presentation of link text.

Discussion
Link text attributes are applied as temporary attributes to any text with a link attribute. Candidates include those attributes that do not affect layout.

In applications created prior to Mac OS X v10.3, the default value is an empty dictionary. In applications created with Mac OS X v10.3 or greater, the default attributes specify blue text with a single underline and the pointing hand cursor.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `setLinkTextAttributes:` (page 72)

Declared In
`NSTextView.h`

loosenKerning:

Increases the space between glyphs in the receiver's selection, or in all text if the receiver is a plain text view.

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

Parameters

sender

The control that sent the message; may be `nil`.

Discussion

Kerning values are determined by the point size of the fonts in the selection.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `tightenKerning:` (page 91)
- `turnOffKerning:` (page 94)
- `useStandardKerning:` (page 99)

Declared In

`NSTextView.h`

lowerBaseline:

Lowers the baseline offset of selected text by 1 point, or of all text if the receiver is a plain text view.

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

Parameters

sender

The control that sent the message; may be `nil`.

Discussion

As such, this method defines a more primitive operation than subscripting.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `raiseBaseline:` (page 48)
- `subscript:` (NSText)
- `unscript:` (NSText)

Declared In

`NSTextView.h`

markedTextAttributes

Returns the attributes used to draw marked text.

```
- (NSDictionary *)markedTextAttributes
```

Return Value

A dictionary of attributes used to draw marked text. Text color, background color, and underline are the only supported attributes for marked text.

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

See Also
`- setMarkedTextAttributes:` (page 72)

Declared In
`NSTextView.h`

orderFrontLinkPanel:

Brings forward a panel allowing the user to manipulate links in the text view.

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

Parameters

sender
The control sending the message. May be `nil`.

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

Declared In
`NSTextView.h`

orderFrontListPanel:

Brings forward a panel allowing the user to manipulate text lists in the text view.

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

Parameters

sender
The control sending the message. May be `nil`.

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

Declared In
`NSTextView.h`

orderFrontSpacingPanel:

Brings forward a panel allowing the user to manipulate text line heights, interline spacing, and paragraph spacing, in the text view.

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

Parameters

sender
The control sending the message. May be `nil`.

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

Declared In
`NSTextView.h`

orderFrontTablePanel:

Brings forward a panel allowing the user to manipulate text tables in the text view.

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

Parameters

sender

The control sending the message. May be `nil`.

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

Declared In
`NSTextView.h`

outline:

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

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

Parameters

sender

The control that sent the message; may be `nil`.

Discussion
If there is a selection and the first character of the selected range has a non-zero stroke width, or if there is no selection and the typing attributes have a non-zero stroke width, then the stroke width is removed; otherwise the value of `NSStrokeWidthAttributeName` is set to the default value for outline (3.0).

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 v10.3 and later.

Declared In
`NSTextView.h`

pasteAsPlainText:

Inserts the contents of the pasteboard into the receiver's text as plain text.

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

Parameters

sender

The control that sent the message; may be `nil`.

Discussion

This method behaves analogously to `insertText:` (page 38).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `pasteAsRichText:` (page 47)
- `insertText:` (page 38)

Declared In

`NSTextView.h`

pasteAsRichText:

This action method inserts the contents of the pasteboard into the receiver's text as rich text, maintaining its attributes.

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

Parameters

sender

The control that sent the message; may be `nil`.

Discussion

The text is inserted at the insertion point if there is one, otherwise replacing the selection.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `pasteAsRichText:` (page 47)
- `insertText:` (page 38)

Declared In

`NSTextView.h`

performFindPanelAction:

Performs a find panel action specified by the sender's tag.

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

Parameters

sender

The control sending the message. This method sends the `tag` method to determine what operation to perform. The list of possible tags is provided in "Constants" (page 114).

Discussion
This is the generic action method for the find menu and find panel, and can be overridden to implement a custom find panel.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTextView.h

preferredPasteboardTypeFromArray:restrictedToTypesFromArray:

Returns whatever type on the pasteboard would be most preferred for copying data.

```
- (NSString *)preferredPasteboardTypeFromArray:(NSArray *)availableTypes
    restrictedToTypesFromArray:(NSArray *)allowedTypes
```

Parameters

availableTypes

The types currently available on the pasteboard.

allowedTypes

Types allowed in the return value. If `nil`, any available type is allowed.

Return Value
The preferred type to provide given the available types and the allowed types.

Discussion
You should not need to override this method. You should also not need to invoke it unless you are implementing a new type of pasteboard to handle services other than copy/paste or dragging.

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

See Also
- `pasteAsPlainText:` (page 46)
- `pasteAsRichText:` (page 47)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
NSTextView.h

raiseBaseline:

Raises the baseline offset of selected text by 1 point, or of all text if the receiver is a plain text view.

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

Parameters

sender

The control that sent the message; may be `nil`.

Discussion

As such, this method defines a more primitive operation than superscripting.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `lowerBaseline:` (page 44)
- `superscript:` (NSText)
- `unscript:` (NSText)

Declared In

`NSTextView.h`

rangeForUserCharacterAttributeChange

Returns the range of characters affected by an action method that changes character (not paragraph) attributes.

```
- (NSRange)rangeForUserCharacterAttributeChange
```

Return Value

The range of characters affected by an action method that changes character (not paragraph) attributes, such as the NSText action method `changeFont:`. For rich text this range is typically the range of the selection. For plain text this range is the entire contents of the receiver. If the receiver isn't editable or doesn't use the Font panel, the range returned has a location of `NSNotFound`.

Special Considerations

In Mac OS X v10.4 and later, returns the first subrange where there is a multiple-range selection.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `rangesForUserCharacterAttributeChange` (page 51)
- `rangeForUserParagraphAttributeChange` (page 50)
- `rangeForUserTextChange` (page 51)
- `isEditable` (page 40)
- `usesFontPanel` (page 98)

Declared In

`NSTextView.h`

rangeForUserCompletion

Returns the partial range from the most recent beginning of a word up to the insertion point.

```
- (NSRange)rangeForUserCompletion
```

Return Value
The partial range from the most recent beginning of a word up to the insertion point. Returning (`NSNotFound`, 0) suppresses completion.

Discussion
May be overridden by subclasses to alter the range to be completed.

The return value from this method is intended to be used for the range argument in the text completion methods such as `completionsForPartialWordRange:indexOfSelectedItem:` (page 29).

Special Considerations

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSTextView.h`

rangeForUserParagraphAttributeChange

Returns the range of characters affected by an action method that changes paragraph (not character) attributes.

```
- (NSRange)rangeForUserParagraphAttributeChange
```

Return Value
The range of characters affected by an action method that changes paragraph (not character) attributes, such as the NSText action method `alignLeft:`. For rich text this range is typically calculated by extending the range of the selection to paragraph boundaries. For plain text this range is the entire contents of the receiver. If the receiver isn't editable or doesn't use the Font panel, the range returned has a location of `NSNotFound`.

Special Considerations

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

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

See Also
- `rangesForUserParagraphAttributeChange` (page 52)
- `rangeForUserCharacterAttributeChange` (page 49)
- `rangeForUserTextChange` (page 51)
- `isEditable` (page 40)
- `usesRuler` (page 98)

Declared In
`NSTextView.h`

rangeForUserTextChange

Returns the range of characters affected by a method that changes characters (as opposed to attributes).

```
- (NSRange)rangeForUserTextChange
```

Return Value
The range of characters affected by a method that changes characters (as opposed to attributes), such as `insertText:` (page 38). This is typically the range of the selection. If the receiver isn't editable the range returned has a location of `NSNotFound`.

Special Considerations

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

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

See Also
- `rangesForUserTextChange` (page 52)
- `rangeForUserParagraphAttributeChange` (page 50)
- `rangeForUserCharacterAttributeChange` (page 49)
- `isEditable` (page 40)
- `usesRuler` (page 98)

Declared In
`NSTextView.h`

rangesForUserCharacterAttributeChange

Returns an array containing the ranges of characters affected by an action method that changes character (not paragraph) attributes.

```
- (NSArray *)rangesForUserCharacterAttributeChange
```

Return Value
An array containing the ranges of characters affected by an action method that changes character (not paragraph) attributes, such as the NSText action method `changeFont:`. For rich text these ranges are typically the ranges of the selections. For plain text the range is the entire contents of the receiver. Returns `nil` if the receiver isn't editable or doesn't use the Font panel.

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

See Also
- `rangesForUserParagraphAttributeChange` (page 52)
- `rangesForUserTextChange` (page 52)
- `isEditable` (page 40)
- `usesFontPanel` (page 98)

Declared In
`NSTextView.h`

rangesForUserParagraphAttributeChange

Returns an array containing the ranges of characters affected by a method that changes paragraph (not character) attributes.

```
- (NSArray *)rangesForUserParagraphAttributeChange
```

Return Value
An array containing the ranges of characters affected by an action method that changes paragraph (not character) attributes, such as the NSText action method `alignLeft:`. For rich text these ranges are typically calculated by extending the range of the selection to paragraph boundaries. For plain text the range is the entire contents of the receiver. Returns `nil` if the receiver isn't editable or doesn't use the Font panel.

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

See Also
- `rangesForUserCharacterAttributeChange` (page 51)
- `rangesForUserTextChange` (page 52)
- `isEditable` (page 40)
- `usesRuler` (page 98)

Declared In
`NSTextView.h`

rangesForUserTextChange

Returns an array containing the ranges of characters affected by a method that changes characters (as opposed to attributes).

```
- (NSArray *)rangesForUserTextChange
```

Return Value
An array containing the ranges of characters affected by a method that changes characters (as opposed to attributes), such as `insertText:` (page 38). These are typically the ranges of the selections. Returns `nil` if the receiver isn't editable.

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

See Also
- `rangesForUserCharacterAttributeChange` (page 51)
- `rangesForUserParagraphAttributeChange` (page 52)
- `isEditable` (page 40)
- `usesRuler` (page 98)

Declared In
`NSTextView.h`

readablePasteboardTypes

Returns the types this text view can read immediately from the pasteboard.

```
- (NSArray *)readablePasteboardTypes
```

Return Value
An array of strings describing the types this text view can read immediately from the pasteboard. The strings are ordered by the default preferences.

Discussion
You can override this method to provide support for new types of data. If you want to add support for the default types, you can invoke the superclass version of this method or add the types directly in your overridden version.

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

See Also
- `preferredPasteboardTypeFromArray:restrictedToTypesFromArray:` (page 48)
- `writablePasteboardTypes` (page 100)

Declared In
`NSTextView.h`

readSelectionFromPasteboard:

Reads the text view's preferred type of data from the specified pasteboard.

```
- (BOOL)readSelectionFromPasteboard:(NSPasteboard *)pboard
```

Parameters
pboard
The pasteboard to read from.

Return Value
`YES` if the data was successfully read, `NO` otherwise.

Discussion
This method invokes the `preferredPasteboardTypeFromArray:restrictedToTypesFromArray:` (page 48) method to determine the text view's preferred type of data and then reads the data using the `readSelectionFromPasteboard:type:` (page 54) method.

You should not need to override this method. You might need to invoke this method if you are implementing a new type of pasteboard to handle services other than copy/paste or dragging.

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

See Also
- `preferredPasteboardTypeFromArray:restrictedToTypesFromArray:` (page 48)
- `readSelectionFromPasteboard:type:` (page 54)

Declared In
`NSTextView.h`

readSelectionFromPasteboard:type:

Reads data of the given type from the specified pasteboard.

```
- (BOOL)readSelectionFromPasteboard:(NSPasteboard *)pboard type:(NSString *)type
```

Parameters

pboard

The pasteboard to read from.

type

The type of data to read.

Return Value

`YES` if the data was successfully read, `NO` otherwise.

Discussion

The new data is placed at the current insertion point, replacing the current selection if one exists.

You should override this method to read pasteboard types other than the default types. Use the `rangeForUserTextChange` (page 51) method to obtain the range of characters (if any) to be replaced by the new data.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `rangeForUserTextChange` (page 51)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In

`NSTextView.h`

replaceTextContainer:

Replaces the text container for the group of text system objects containing the receiver, keeping the association between the receiver and its layout manager intact.

```
- (void)replaceTextContainer:(NSTextContainer *)aTextContainer
```

Parameters

aTextContainer

The new text container. This method raises `NSInvalidArgumentException` if *aTextContainer* is `nil`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `initWithFrame:textContainer:` (page 36)
- `setTextContainer:` (page 79)

Declared In
NSTextView.h

rulerView:didAddMarker:

Modifies the paragraph style of the paragraphs containing the selection to accommodate a new marker.

```
- (void)rulerView:(NSRulerView *)aRulerView didAddMarker:(NSRulerMarker *)aMarker
```

Parameters

aRulerView
The ruler view sending the message.

aMarker
The marker that was added.

Discussion
This method records the change by invoking `didChangeText` (page 31) after adding the marker.

`NSTextView` checks for permission to make the change in its `rulerView:shouldAddMarker:` (page 57) method, which invokes `shouldChangeTextInRange:replacementString:` (page 83) to send out the proper request and notifications, and only invokes this method if permission is granted.

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

See Also
- `representedObject` (NSRulerMarker)
- `rulerView:didMoveMarker:` (page 55)
- `rulerView:didRemoveMarker:` (page 56)

Declared In
NSTextView.h

rulerView:didMoveMarker:

Modifies the paragraph style of the paragraphs containing the selection to record the new location of the marker.

```
- (void)rulerView:(NSRulerView *)aRulerView didMoveMarker:(NSRulerMarker *)aMarker
```

Parameters

aRulerView
The ruler view sending the message.

aMarker
The marker that was moved.

Discussion
This method records the change by invoking `didChangeText` (page 31) after moving the marker.

`NSTextView` checks for permission to make the change in its `rulerView:shouldMoveMarker:` (page 57) method, which invokes `shouldChangeTextInRange:replacementString:` (page 83) to send out the proper request and notifications, and only invokes this method if permission is granted.

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

See Also
- `representedObject` (NSRulerMarker)
- `rulerView:didAddMarker:` (page 55)
- `rulerView:didRemoveMarker:` (page 56)

Declared In
`NSTextView.h`

rulerView:didRemoveMarker:

Modifies the paragraph style of the paragraphs containing the selection—if possible—by removing the specified marker.

```
- (void)rulerView:(NSRulerView *)aRulerView didRemoveMarker:(NSRulerMarker *)aMarker
```

Parameters

aRulerView
The ruler view sending the message.

aMarker
The marker that was removed.

Discussion
This method records the change by invoking `didChangeText` (page 31) after removing the marker.

`NSTextView` checks for permission to move or remove a tab stop in its `rulerView:shouldMoveMarker:` (page 57) method, which invokes `shouldChangeTextInRange:replacementString:` (page 83) to send out the proper request and notifications, and only invokes this method if permission is granted.

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

See Also
- `representedObject` (NSRulerMarker)
- `shouldChangeTextInRange:replacementString:` (page 83)
- `rulerView:didAddMarker:` (page 55)
- `rulerView:didMoveMarker:` (page 55)

Declared In
`NSTextView.h`

rulerView:handleMouseDown:

Adds a left tab marker to the ruler at the location clicked.

```
- (void)rulerView:(NSRulerView *)aRulerView handleMouseDown:(NSEvent *)theEvent
```

Parameters

aRulerView

The ruler view sending the message.

theEvent

The mouse down event.

Discussion

A subclass can override this method to provide other behavior, such as creating guidelines. This method is invoked once with *theEvent* when the user first clicks the ruler area of *aRulerView*, as described in the NSRulerView class specification.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSTextView.h

rulerView:shouldAddMarker:

Returns whether a new marker can be added.

```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldAddMarker:(NSRulerMarker *)aMarker
```

Parameters

aRulerView

The ruler view sending the message.

aMarker

The marker to be added.

Return Value

YES if *aMarker* can be added, NO otherwise.

Discussion

The receiver checks for permission to make the change by invoking shouldChangeTextInRange:replacementString: (page 83) and returning the return value of that message. If the change is allowed, the receiver is then sent a rulerView:didAddMarker: (page 55) message.

Availability

Available in Mac OS X v10.0 and later.

See Also

- rulerView:shouldMoveMarker: (page 57)
- rulerView:shouldRemoveMarker: (page 58)

Declared In

NSTextView.h

rulerView:shouldMoveMarker:

Returns whether the marker should be moved.

```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldMoveMarker:(NSRulerMarker *)aMarker
```

Parameters

aRulerView

The ruler view sending the message.

aMarker

The marker to be moved.

Return Value

YES if *aMarker* can be moved, NO otherwise.

Discussion

This method controls whether an existing marker *aMarker* can be moved. The receiver checks for permission to make the change by invoking shouldChangeTextInRange:replacementString: (page 83) and returning the return value of that message. If the change is allowed, the receiver is then sent a rulerView:didMoveMarker: (page 55) message.

Availability

Available in Mac OS X v10.0 and later.

See Also

- rulerView:shouldAddMarker: (page 57)
- rulerView:shouldRemoveMarker: (page 58)

Declared In

NSTextView.h

rulerView:shouldRemoveMarker:

Returns whether the marker should be removed.

```
- (BOOL)rulerView:(NSRulerView *)aRulerView shouldRemoveMarker:(NSRulerMarker
    *)aMarker
```

Parameters

aRulerView

The ruler view sending the message.

aMarker

The marker to be removed.

Return Value

YES if *aMarker* can be removed, NO otherwise.

Discussion

Only markers that represent tab stops can be removed. This method returns YES if *aMarker* represents an NSTextTab object, NO otherwise. Because this method can be invoked repeatedly as the user drags a ruler marker, it returns that value immediately. If the change is allowed and the user actually removes the marker, the receiver is also sent a rulerView:didRemoveMarker: (page 56) message.

Availability

Available in Mac OS X v10.0 and later.

See Also

- rulerView:shouldAddMarker: (page 57)
- rulerView:shouldMoveMarker: (page 57)

Declared In
NSTextView.h

rulerView:willAddMarker:atLocation:

Returns a potentially modified location to which the marker should be added.

```
- (CGFloat)rulerView:(NSRulerView *)aRulerView willAddMarker:(NSRulerMarker *)aMarker
    atLocation:(CGFloat)location
```

Parameters

aRulerView
The ruler view sending the message.

aMarker
The marker to be added.

location
The new location for the marker.

Return Value
The modified location to which the marker should be added.

Discussion
This method ensures that the proposed *location* of *aMarker* lies within the appropriate bounds for the receiver's text container, returning the modified location.

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

See Also
- rulerView:didAddMarker: (page 55)

Declared In
NSTextView.h

rulerView:willMoveMarker:toLocation:

Returns a potentially modified location to which the marker should be moved.

```
- (CGFloat)rulerView:(NSRulerView *)aRulerView willMoveMarker:(NSRulerMarker
    *)aMarker toLocation:(CGFloat)location
```

Parameters

aRulerView
The ruler view sending the message.

aMarker
The marker to be moved.

location
The new location for the marker.

Return Value
The modified location to which the marker should be moved.

Discussion
This method ensures that the proposed *location* of *aMarker* lies within the appropriate bounds for the receiver's text container.

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

See Also
- rulerView:didMoveMarker: (page 55)

Declared In
NSTextView.h

selectedRanges

Returns an array containing the ranges of characters selected in the receiver's layout manager.

```
- (NSArray *)selectedRanges
```

Return Value
A non-nil, non-empty array of objects responding to the NSValue rangeValue method. In addition, the objects in the array are sorted, non-overlapping, non-contiguous, and (except for the case of a single range) have non-zero-length.

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

See Also
- setSelectedRanges: (page 76)

Declared In
NSTextView.h

selectedTextAttributes

Returns the attributes used to indicate the selection.

```
- (NSDictionary *)selectedTextAttributes
```

Return Value
A dictionary of attributes used to indicate the selection. Text color, background color, and underline are the only supported attributes for selected text. Typically only the text background color is used.

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

See Also
- selectedRange (NSTextInput)
- setSelectedTextAttributes: (page 77)

Declared In
NSTextView.h

selectionAffinity

Returns the preferred direction of selection.

```
- (NSSelectionAffinity)selectionAffinity
```

Return Value
The preferred direction of selection.

Discussion
Selection affinity determines whether, for example, the insertion point appears after the last character on a line or before the first character on the following line in cases where text wraps across line boundaries.

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

See Also
- `setSelectedRange:affinity:stillSelecting:` (page 75)

Declared In
`NSTextView.h`

selectionGranularity

Returns the current selection granularity, used during mouse tracking to modify the range of the selection.

```
- (NSSelectionGranularity)selectionGranularity
```

Return Value
The current selection granularity.

Discussion
See `setSelectionGranularity:` (page 78) for a discussion of how selection granularity affects the behavior of selection extension.

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

See Also
- `selectionRangeForProposedRange:granularity:` (page 61)
- `setSelectionGranularity:` (page 78)

Declared In
`NSTextView.h`

selectionRangeForProposedRange:granularity:

Returns an adjusted selected range based on the selection granularity.

```
- (NSRange)selectionRangeForProposedRange:(NSRange)proposedSelRange
    granularity:(NSSelectionGranularity)granularity
```

Parameters

proposedSelRange

The proposed selected range.

granularity

The selection granularity.

Return Value

The adjusted selected range, taking into account the selection granularity.

Discussion

This method is invoked repeatedly during mouse tracking to modify the range of the selection. Override this method to specialize selection behavior.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setSelectionGranularity:` (page 78)

Declared In

`NSTextView.h`

setAcceptsGlyphInfo:

Sets whether the receiver accepts the glyph info attribute.

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

Parameters

flag

`YES` if the receiver should accept the `NSGlyphInfoAttributeName` attribute from text input sources such as input methods and the pasteboard, `NO` otherwise.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `acceptsGlyphInfo` (page 23)

Declared In

`NSTextView.h`

setAlignment:range:

Sets the alignment of the paragraphs containing characters in the specified range.

```
- (void)setAlignment:(NSTextAlignment)alignment range:(NSRange)aRange
```

Parameters

alignment

The new alignment.

aRange

The range of characters whose paragraphs will have their alignment set.

Discussion
This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 84) or `shouldChangeTextInRange:replacementString:` (page 83) to include this method in an undoable action.

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

See Also
`- rangeForUserParagraphAttributeChange` (page 50)

Declared In
`NSTextView.h`

setAllowedInputSourceLocales:

Sets an array of locale identifiers representing input sources that are allowed to be enabled when the receiver has the keyboard focus.

```
- (void)setAllowedInputSourceLocales:(NSArray *)localeIdentifiers
```

Parameters

localeIdentifiers

The new locale identifiers of allowed input sources.

Discussion
You can use the meta-locale identifier, `NSAllRomanInputSourcesLocaleIdentifier` (page 117), to specify input sources that are limited for Roman script editing.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- allowedInputSourceLocales` (page 24)

Declared In
`NSTextView.h`

setAllowsDocumentBackgroundColorChange:

Sets whether the receiver allows its background color to change.

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

Parameters

flag

`YES` if the receiver allows the background color to change, otherwise `NO`.

Discussion
This corresponds to the background color of the entirety of the text view, not just to a selected range of text.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `allowsDocumentBackgroundColorChange` (page 24)
- `changeDocumentBackgroundColor:` (page 27)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
`NSTextView.h`

setAllowsImageEditing:

Specifies whether image attachments should permit editing of their images.

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

Parameters
flag
If `YES`, image editing is allowed; if `NO`, it is not allowed.

Discussion
For image editing to be allowed, the text view must be editable and the text attachment cell must support image editing.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `allowsImageEditing` (page 25)

Declared In
`NSTextView.h`

setAllowsUndo:

Sets whether undo support is enabled.

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

Parameters
flag
`YES` to enable undo support, `NO` otherwise.

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

See Also
- `allowsUndo` (page 25)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
Sketch-112
TextEditPlus

Declared In
`NSTextView.h`

setAutomaticLinkDetectionEnabled:

Enables or disables automatic link detection.

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

Parameters

flag

If `YES`, automatic link detection is enabled; if `NO`, it is disabled.

Discussion
Automatic link detection causes strings representing URLs typed in the view to be automatically made into links to those URLs.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- isAutomaticLinkDetectionEnabled` (page 39)
`- toggleAutomaticLinkDetection:` (page 91)
`URLAtIndex:effectiveRange:` (`NSAttributedString`)

Declared In
`NSTextView.h`

setAutomaticQuoteSubstitutionEnabled:

Enables and disables automatic quotation mark substitution.

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

Parameters

flag

If `YES`, automatic quotation mark substitution is enabled; if `NO`, it is disabled.

Discussion
Automatic quote substitution causes ASCII quotation marks and apostrophes to be automatically replaced, on a context-dependent basis, with more typographically accurate symbols.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- isAutomaticQuoteSubstitutionEnabled` (page 39)
`- toggleAutomaticQuoteSubstitution:` (page 92)

Declared In
NSTextView.h

setBackgroundColor:

Sets the receiver's background color.

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

Parameters

aColor

The new background color.

Special Considerations

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 84) or `shouldChangeTextInRange:replacementString:` (page 83) to include this method in an undoable action.

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

See Also
- `setDrawsBackground:` (page 69)
- `backgroundColor` (page 26)

Declared In
NSTextView.h

setBaseWritingDirection:range:

Sets the base writing direction of a range of text.

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

Parameters

writingDirection

The new writing direction for the text in *range*.

range

The range of text that will have the new writing direction.

Discussion
Invoke this method to change the base writing direction from left-to-right to right-to-left for languages like Hebrew and Arabic, for example.

This method does not include undo support by default. Clients must invoke `shouldChangeTextInRanges:replacementStrings:` (page 84) or `shouldChangeTextInRange:replacementString:` (page 83) to include this method in an undoable action.

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

Declared In
NSTextView.h

setConstrainedFrameSize:

Attempts to set the frame size as if by user action.

```
- (void)setConstrainedFrameSize:(NSSize)desiredSize
```

Parameters

desiredSize
The new desired size.

Discussion
This method respects the receiver's existing minimum and maximum sizes and by whether resizing is permitted.

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

See Also
- `minSize` (NSText)
- `maxSize` (NSText)
- `isHorizontallyResizable` (NSText)
- `isVerticallyResizable` (NSText)

Declared In
NSTextView.h

setContinuousSpellCheckingEnabled:

Enables or disables continuous spell checking.

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

Parameters

flag
If `YES`, enables continuous spell checking; if `NO`, disables it.

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

See Also
- `isContinuousSpellCheckingEnabled` (page 40)
- `toggleContinuousSpellChecking:` (page 93)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
NSTextView.h

setDefaultParagraphStyle:

Sets the receiver's default paragraph style.

```
- (void)setDefaultParagraphStyle:(NSParagraphStyle *)paragraphStyle
```

Parameters

paragraphStyle

The new default paragraph style.

Availability

Available in Mac OS X v10.3 and later.

See Also

- defaultParagraphStyle (page 30)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In

NSTextView.h

setDelegate:

Sets the delegate for all text views sharing the receiver's layout manager.

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

Parameters

anObject

The new delegate object.

Availability

Available in Mac OS X v10.0 and later.

See Also

- delegate (page 30)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Sketch-112

TextEditPlus

Declared In

NSTextView.h

setDisplaysLinkToolTips:

Enables or disables automatic display of link tooltips.

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

Parameters

flag

If YES, automatic link tooltip display is enabled; if NO, it is disabled.

Discussion

The default value for this feature is YES; clients who do not wish tooltips to be displayed automatically must explicitly disable it.

Availability

Available in Mac OS X v10.5 and later.

See Also

- displaysLinkToolTips (page 31)

Declared In

NSTextView.h

setDrawsBackground:

Sets whether the receiver draws its background.

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

Parameters

flag

YES to cause the receiver to fill its background with the background color, NO otherwise.

Availability

Available in Mac OS X v10.0 and later.

See Also

- setBackgroundColor: (page 66)
- drawsBackground (page 34)

Related Sample Code

Sketch-112

Declared In

NSTextView.h

setEditable:

Controls whether the text views sharing the receiver's layout manager allow the user to edit text.

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

Parameters

flag

YES to allow the user to edit text and attributes of all text views sharing the receiver's layout manager, NO otherwise.

Discussion

If a text view is made editable, it's also made selectable. Text views are editable by default.

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

See Also
- `setSelectable:` (page 74)
- `isEditable` (page 40)

Declared In
`NSTextView.h`

setFieldEditor:

Controls whether the text views sharing the receiver's layout manager behave as field editors.

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

Parameters

flag

`YES` to cause the text views sharing the receiver's layout manager to behave as field editors, `NO` otherwise.

Discussion
Field editors interpret Tab, Shift-Tab, and Return (Enter) as cues to end editing and possibly to change the first responder. Non-field editors instead accept these characters as text input. See Text Fields, Text Views, and the Field Editor for more information on field editors. By default, text views don't behave as field editors.

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

See Also
- `isFieldEditor` (page 40)

Declared In
`NSTextView.h`

setGrammarCheckingEnabled:

Enables and disables grammar checking.

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

Parameters

flag

If `YES`, grammar checking is enabled; if `NO`, it is disabled.

Discussion
If grammar checking is enabled, then it is performed alongside spell checking, whenever the text view checks spelling, whether continuously or manually.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `isGrammarCheckingEnabled` (page 41)
- `toggleGrammarChecking:` (page 93)
- `setSpellingState:range:` (page 79)

Declared In
`NSTextView.h`

setImportsGraphics:

Controls whether the text views sharing the receiver's layout manager allow the user to import files by dragging.

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

Parameters

flag

`YES` to allow the user to import files by dragging onto the text views sharing the receiver's layout manager, `NO` otherwise.

Discussion
Text views that are set to accept dragged files are also set to allow rich text. By default, text views don't accept dragged files but do allow rich text.

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

See Also
- `textStorage` (page 90)
- `setRichText:` (page 73)
- `importsGraphics` (page 35)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
`NSTextView.h`

setInsertionPointColor:

Sets the color of the insertion point

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

Parameters

aColor

The new color.

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

See Also
- `drawInsertionPointInRect:color:turnedOn:` (page 34)
- `shouldDrawInsertionPoint` (page 84)
- `insertionPointColor` (page 37)

Declared In
`NSTextView.h`

setLinkTextAttributes:

Sets the attributes used to draw the onscreen presentation of link text.

```
- (void)setLinkTextAttributes:(NSDictionary *)attributeDictionary
```

Parameters

attributeDictionary
A dictionary of attributes corresponding to the onscreen presentation of link text.

Discussion
Link text attributes are applied as temporary attributes to any text with a link attribute. Candidates include those attributes that do not affect layout.

In applications created prior to Mac OS X v10.3, the default value is an empty dictionary. In applications created with Mac OS X v10.3 or greater, the default attributes specify blue text with a single underline and the pointing hand cursor.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `linkTextAttributes` (page 43)

Declared In
`NSTextView.h`

setMarkedTextAttributes:

Sets the attributes used to draw marked text.

```
- (void)setMarkedTextAttributes:(NSDictionary *)attributes
```

Parameters

attributes
A dictionary of attributes used to draw marked text. Text color, background color, and underline are the only supported attributes for marked text.

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

See Also
- `markedTextAttributes` (page 44)
- `markedRange` (NSTextInput)

Declared In
NSTextView.h

setNeedsDisplayInRect:avoidAdditionalLayout:

Marks the receiver as requiring display.

```
- (void)setNeedsDisplayInRect:(NSRect)aRect avoidAdditionalLayout:(BOOL)flag
```

Parameters

aRect

The rectangle in which display is required.

flag

A value of YES causes the receiver to not perform any layout, even if this means that portions of the text view remain empty. Otherwise the receiver performs at least as much layout as needed to display *aRect*.

Discussion
NSTextView overrides the NSView setNeedsDisplayInRect: method to invoke this method with a *flag* argument of NO.

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

Declared In
NSTextView.h

setRichText:

Controls whether the text views sharing the receiver's layout manager allow the user to apply attributes to specific ranges of text.

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

Parameters

flag

YES to allow the user to apply attributes to specific ranges of text in text views sharing the receiver's layout manager, NO otherwise.

Discussion
Text fields that don't allow rich text also don't accept dragged files. By default, text views let the user apply multiple attributes to text, but don't accept dragged files.

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

See Also
- textStorage (page 90)
- isRichText (page 41)
- setImportsGraphics: (page 71)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit

TextEditPlus
TipWrapper

Declared In
NSTextView.h

setRulerVisible:

Controls whether the scroll view enclosing text views sharing the receiver's layout manager displays the ruler.

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

Parameters

flag

YES to show the ruler, NO to hide the ruler. By default, the ruler is hidden.

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

See Also
- setUsesRuler: (page 82)
- isRulerVisible (page 42)
- toggleRuler: (NSText)

Declared In
NSTextView.h

setSelectable:

Controls whether the text views sharing the receiver's layout manager allow the user to select text.

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

Parameters

flag

YES to allow the user to select text of all text views sharing the receiver's layout manager; otherwise, NO.

Discussion
If a text view is made not selectable, it's also made not editable, and buttons on the Find panel are dimmed. Text views are by default both editable and selectable.

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

See Also
- setEditable: (page 69)
- isSelectable (page 42)

Declared In
NSTextView.h

setSelectedRange:

Sets the selection to the characters in a single range.

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

Parameters

charRange

The range of characters to select. This range must begin and end on glyph boundaries and not split base glyphs and their nonspacing marks.

Discussion

This method sets the selection to the characters in *charRange*, resets the selection granularity to `NSSelectByCharacter`, and posts an `NSTextViewDidChangeSelectionNotification` (page 119) to the default notification center. It also removes the marking from marked text if the new selection is greater than the marked region.

Special Considerations

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setSelectedRange:affinity:stillSelecting:` (page 75)
- `selectionAffinity` (page 61)
- `selectionGranularity` (page 61)
- `selectedRange` (NSTextInput)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Sketch-112

TextEditPlus

TextViewDelegate

WebKitPluginWithSimpleGUI

Declared In

`NSTextView.h`

setSelectedRange:affinity:stillSelecting:

Sets the selection to a range of characters in response to user action.

```
- (void)setSelectedRange:(NSRange)charRange affinity:(NSSelectionAffinity)affinity
    stillSelecting:(BOOL)flag
```

Parameters

charRange

The range of characters to select. This range must begin and end on glyph boundaries and not split base glyphs and their nonspacing marks.

affinity

The selection affinity for the selection. See `selectionAffinity` (page 61) for more information about how affinities work.

flag

`YES` to behave appropriately for a continuing selection where the user is still dragging the mouse, `NO` otherwise. If `YES`, the receiver doesn't send notifications or remove the marking from its marked text. If `NO`, the receiver posts an `NSTextViewDidChangeSelectionNotification` (page 119) to the default notification center and removes the marking from marked text if the new selection is greater than the marked region.

Discussion

This method resets the selection granularity to `NSSelectByCharacter`.

Special Considerations

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setSelectedRange:` (page 75)
- `selectionAffinity` (page 61)
- `selectionGranularity` (page 61)
- `selectedRange` (NSTextInput)

Declared In

`NSTextView.h`

setSelectedRanges:

Sets the selection to the characters in an array of ranges.

```
- (void)setSelectedRanges:(NSArray *)ranges
```

Parameters

ranges

A non-nil, non-empty array of objects responding to the NSValue `rangeValue` method. The ranges in the *ranges* array must begin and end on glyph boundaries and not split base glyphs and their nonspacing marks.

Discussion

Sets the selection to the characters in the *ranges* array, resets the selection granularity to `NSSelectByCharacter`, and posts an `NSTextViewDidChangeSelectionNotification` (page 119) to the default notification center. Also removes the marking from marked text if the new selection is greater than the marked region.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `selectedRanges` (page 60)
- `setSelectedRanges:affinity:stillSelecting:` (page 77)

- selectionAffinity (page 61)
- selectionGranularity (page 61)

Declared In
NSTextView.h

setSelectedRanges:affinity:stillSelecting:

Sets the selection to the characters in an array of ranges in response to user action.

```
- (void)setSelectedRanges:(NSArray *)ranges affinity:(NSSelectionAffinity)affinity
    stillSelecting:(BOOL)stillSelectingFlag
```

Parameters

ranges

A non-nil, non-empty array of objects responding to the NSValue rangeValue method. The ranges in the *ranges* array must begin and end on glyph boundaries and not split base glyphs and their nonspacing marks.

affinity

The selection affinity for the selection. See selectionAffinity (page 61) for more information about how affinities work.

stillSelectingFlag

YES to behave appropriately for a continuing selection where the user is still dragging the mouse, NO otherwise. If YES, the receiver doesn't send notifications or remove the marking from its marked text. If NO, the receiver posts an NSTextViewDidChangeSelectionNotification (page 119) to the default notification center and removes the marking from marked text if the new selection is greater than the marked region.

Discussion
This method also resets the selection granularity to NSSelectByCharacter.

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

See Also
- selectedRanges (page 60)
- setSelectedRanges: (page 76)
- selectionAffinity (page 61)
- selectionGranularity (page 61)

Declared In
NSTextView.h

setSelectedTextAttributes:

Sets the attributes used to indicate the selection.

```
- (void)setSelectedTextAttributes:(NSDictionary *)attributes
```

Parameters

attributes

A dictionary of attributes used to indicate the selection. Text color, background color, and underline are the only supported attributes for selected text.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `selectedRange` (NSTextInput)
- `selectedTextAttributes` (page 60)

Declared In

`NSTextView.h`

setSelectionGranularity:

Sets the selection granularity for subsequent extension of a selection.

```
- (void)setSelectionGranularity:(NSSelectionGranularity)granularity
```

Parameters

granularity

The new granularity for selection extension.

Discussion

Selection granularity is used to determine how the selection is modified when the user Shift-clicks or drags the mouse after a double or triple click. For example, if the user selects a word by double-clicking, the selection granularity is set to `NSSelectByWord`. Subsequent Shift-clicks then extend the selection by words.

Selection granularity is reset to `NSSelectByCharacter` whenever the selection is set. You should always set the selection granularity after setting the selection.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `selectionGranularity` (page 61)
- `setSelectedRange:` (page 75)

Declared In

`NSTextView.h`

setSmartInsertDeleteEnabled:

Controls whether the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation.

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

Parameters

flag

YES if the receiver should insert or delete space around selected words so as to preserve proper spacing and punctuation, NO if it should insert and delete exactly what's selected.

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

See Also
- smartInsertForString:replacingRange:beforeString:afterString: (page 87)
- smartDeleteRangeForProposedRange: (page 85)
- smartInsertDeleteEnabled (page 87)

Declared In
NSTextView.h

setSpellingState:range:

Sets the spelling state, which controls the display of the spelling and grammar indicators on the given text range.

```
- (void)setSpellingState:(NSInteger)value range:(NSRange)charRange
```

Parameters

value

The spelling state value to set. Possible values, for the temporary attribute on the layout manager using the key NSSpellingStateAttributeName, are:

NSSpellingStateSpellingFlag to highlight spelling issues.

NSSpellingStateGrammarFlag to highlight grammar issues.

charRange

The character range over which to set the given spelling state.

Discussion
May be called or overridden to control setting of spelling and grammar indicators on text, used to highlight portions of the text that are flagged for spelling or grammar issues.

Calls the delegate method textView:shouldSetSpellingState:range: (page 109).

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTextView.h

setTextContainer:

Sets the receiver's text container.

```
- (void)setTextContainer:(NSTextContainer *)aTextContainer
```

Parameters

aTextContainer

The new text container.

Discussion

The receiver uses the layout manager and text storage of *aTextContainer*.

Special Considerations

This method is invoked automatically when you create a text view; you should never invoke it directly, but might want to override it. To change the text view for an established group of text system objects, send `setTextView:` to the text container. To replace the text container for a text view and maintain the view's association with the existing layout manager and text storage, use `replaceTextContainer:` (page 54).

Availability

Available in Mac OS X v10.0 and later.

See Also

- `textContainer` (page 89)

Declared In

`NSTextView.h`

setTextContainerInset:

Sets the empty space the receiver leaves around its associated text container.

```
- (void)setTextContainerInset:(NSSize)inset
```

Parameters

inset

The empty space to leave around the text view's text container.

Discussion

It is possible to set the text container and view sizes and resizing behavior so that the inset cannot be maintained exactly, although the text system tries to maintain the inset wherever possible. In any case, the `textContainerOrigin` (page 90) and size of the text container are authoritative as to the location of the text container within the view.

The text itself can have an additional inset, inside the text container, specified by the `setLineFragmentPadding:` method of `NSTextContainer`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `textContainerOrigin` (page 90)
- `invalidateTextContainerOrigin` (page 38)
- `textContainerInset` (page 90)

Related Sample Code

Sketch-112

Declared In

`NSTextView.h`

setTypingAttributes:

Sets the receiver's typing attributes.

```
- (void)setTypingAttributes:(NSDictionary *)attributes
```

Parameters

attributes

A dictionary of the new typing attributes.

Discussion

Typing attributes are reset automatically whenever the selection changes. However, if you add any user actions that change text attributes, the action should use this method to apply those attributes afterwards. User actions that change attributes should always set the typing attributes because there might not be a subsequent change in selection before the next typing.

Availability

Available in Mac OS X v10.0 and later.

See Also

- typingAttributes (page 95)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In

NSTextView.h

setUsesFindPanel:

Specifies whether the receiver allows for a find panel.

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

Parameters

flag

YES to allow the use of a find panel, NO otherwise.

Availability

Available in Mac OS X v10.3 and later.

See Also

- usesFindPanel (page 97)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In

NSTextView.h

setUsesFontPanel:

Controls whether the text views sharing the receiver's layout manager use the Font panel and Font menu.

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

Parameters

flag

YES to make the text views sharing the receiver's layout manager respond to messages from the Font panel and from the Font menu, and update the Font panel with the selection font whenever it changes, NO to disallow character attribute changes.

Discussion
By default, text view objects use the Font panel and menu.

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

See Also
- rangeForUserCharacterAttributeChange (page 49)
- usesFontPanel (page 98)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
NSTextView.h

setUsesRuler:

Controls whether the text views sharing the receiver's layout manager use a ruler.

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

Parameters

flag

YES to cause text views sharing the receiver's layout manager to respond to NSRulerView client messages and to paragraph-related menu actions, and update the ruler (when visible) as the selection changes with its paragraph and tab attributes, otherwise NO.

Discussion
Text views must use a ruler to respond to Format menu commands. If a set of text views don't use the ruler, the ruler is hidden, and the text views disallow paragraph attribute changes. By default, text view objects use the ruler.

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

See Also
- setRulerVisible: (page 74)
- rangeForUserParagraphAttributeChange (page 50)
- usesRuler (page 98)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
NSTextView.h

shouldChangeTextInRange:replacementString:

Initiates a series of delegate messages (and general notifications) to determine whether modifications can be made to the characters and attributes of the receiver's text.

```
- (BOOL)shouldChangeTextInRange:(NSRange)affectedCharRange
    replacementString:(NSString *)replacementString
```

Parameters

affectedCharRange
The range of characters affected by the proposed change.

replacementString
The characters that will replace those in *affectedCharRange*. If only text attributes are being changed, *replacementString* is nil.

Return Value
YES to allow the change, NO to prohibit it.

Discussion
This method checks with the delegate as needed using textShouldBeginEditing: and textView:shouldChangeTextInRange:replacementString: (page 107).

This method must be invoked at the start of any sequence of user-initiated editing changes. If your subclass of NSTextView implements new methods that modify the text, make sure to invoke this method to determine whether the change should be made. If the change is allowed, complete the change by invoking the didChangeText (page 31) method. If you can't determine the affected range or replacement string before beginning changes, pass (NSNotFound, 0) and nil for these values.

Special Considerations

If the receiver is not editable, this method automatically returns NO. This result prevents instances in which a text view could be changed by user actions even though it had been set to be noneditable.

In Mac OS X version 10.4 and later, if there are multiple selections, this method acts on the first selected subrange.

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

See Also
- isEditable (page 40)
- shouldChangeTextInRanges:replacementStrings: (page 84)

Declared In
NSTextView.h

shouldChangeTextInRanges:replacementStrings:

Initiates a series of delegate messages (and general notifications) to determine whether modifications can be made to the characters and attributes of the receiver's text.

```
- (BOOL)shouldChangeTextInRanges:(NSArray *)affectedRanges
    replacementStrings:(NSArray *)replacementStrings
```

Parameters

affectedRanges

An array of ranges to change.

replacementStrings

An array of strings containing the characters that replace those in *affectedRanges*, one for each range. If only text attributes are being changed, *replacementStrings* is `nil`.

Return Value

`YES` to allow the change, `NO` to prohibit it.

Discussion

This method checks with the delegate as needed using `textShouldBeginEditing:` and `textView:shouldChangeTextInRanges:replacementStrings:` (page 108).

This method must be invoked at the start of any sequence of user-initiated editing changes. If your subclass of `NSTextView` implements new methods that modify the text, make sure to invoke this method to determine whether the change should be made. If the change is allowed, complete the change by invoking the `didChangeText` (page 31) method. If you can't determine the affected range or replacement string before beginning changes, pass `nil` for these values.

Special Considerations

If the receiver is not editable, this method automatically returns `NO`. This result prevents instances in which a text view could be changed by user actions even though it had been set to be noneditable.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- isEditable` (page 40)

Declared In

`NSTextView.h`

shouldDrawInsertionPoint

Returns whether the receiver should draw its insertion point.

```
- (BOOL)shouldDrawInsertionPoint
```

Return Value

`YES` if the receiver should draw its insertion point, `NO` otherwise.

Availability

Available in Mac OS X v10.0 and later.

See Also
- drawInsertionPointInRect:color:turnedOn: (page 34)

Declared In
NSTextView.h

showFindIndicatorForRange:

Causes a temporary highlighting effect to appear around the visible portion (or portions) of the specified range.

```
- (void)showFindIndicatorForRange:(NSRange)charRange
```

Parameters

charRange

The character range around which indicators appear.

Discussion
This method supports lozenge-style indication of find results. The indicators automatically disappear after a certain period of time, or when the method is called again, or when any of a number of changes occur to the view (such as changes to text, view size, or view position).

This method does not itself scroll the specified range to be visible; any desired scrolling should be done before this method is called, first, because the method acts only on the visible portion of the specified range, and, second, because scrolling causes the indicators to disappear. Calling this method with a zero-length range always removes any existing indicators.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSTextView.h

smartDeleteRangeForProposedRange:

Returns an extended range that includes adjacent whitespace that should be deleted along with the proposed range in order to preserve proper spacing and punctuation.

```
- (NSRange)smartDeleteRangeForProposedRange:(NSRange)proposedCharRange
```

Parameters

proposedCharRange

The proposed character range for deleting.

Return Value
An extended range that includes adjacent whitespace that should be deleted along with the proposed range in order to preserve proper spacing and punctuation of the text surrounding the deletion.

Discussion
NSTextView uses this method as necessary; you can also use it in implementing your own methods that delete text, typically when the selection granularity is NSSelectByWord. To do so, invoke this method with the proposed range to delete, then actually delete the range returned. If placing text on the pasteboard, however, you should put only the characters from the proposed range onto the pasteboard.

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

See Also
- `smartInsertForString:replacingRange:beforeString:afterString:` (page 87)
- `selectionGranularity` (page 61)
- `smartInsertDeleteEnabled` (page 87)

Declared In
`NSTextView.h`

smartInsertAfterStringForString:replacingRange:

Returns any whitespace that needs to be added after the string to preserve proper spacing and punctuation when the string replaces the characters in the specified range.

```
- (NSString *)smartInsertAfterStringForString:(NSString *)aString
    replacingRange:(NSRange)charRange
```

Parameters

aString
The string that is replacing the characters in *charRange*.

charRange
The range of characters which *aString* is replacing.

Return Value
Any whitespace that needs to be added after *aString* to preserve proper spacing and punctuation when the characters in *charRange* are replaced by *aString*. If *aString* is `nil` or if smart insertion and deletion are disabled, this method returns `nil`.

Discussion
Don't invoke this method directly. Instead, use `smartInsertForString:replacingRange:beforeString:afterString:` (page 87), which calls this method as part of its implementation.

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

Declared In
`NSTextView.h`

smartInsertBeforeStringForString:replacingRange:

Returns any whitespace that needs to be added before the string to preserve proper spacing and punctuation when the string replaces the characters in the specified range.

```
- (NSString *)smartInsertBeforeStringForString:(NSString *)aString
    replacingRange:(NSRange)charRange
```

Parameters

aString
The string that is replacing the characters in *charRange*.

charRange

The range of characters which *aString* is replacing.

Return Value

Any whitespace that needs to be added before *aString* to preserve proper spacing and punctuation when the characters in *charRange* are replaced by *aString*. If *aString* is `nil` or if smart insertion and deletion are disabled, this method returns `nil`.

Discussion

Don't invoke this method directly. Instead, use `smartInsertForString:replacingRange:beforeString:afterString:` (page 87), which calls this method as part of its implementation.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextView.h`

smartInsertDeleteEnabled

Returns whether the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation.

```
- (BOOL)smartInsertDeleteEnabled
```

Return Value

`YES` if the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation, `NO` if it inserts and deletes exactly what's selected.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `smartInsertForString:replacingRange:beforeString:afterString:` (page 87)
- `smartDeleteRangeForProposedRange:` (page 85)
- `setSmartInsertDeleteEnabled:` (page 78)

Declared In

`NSTextView.h`

smartInsertForString:replacingRange:beforeString:afterString:

Determines whether whitespace needs to be added around the string to preserve proper spacing and punctuation when it replaces the characters in the specified range.

```
- (void)smartInsertForString:(NSString *)aString replacingRange:(NSRange)charRange
    beforeString:(NSString **)beforeString afterString:(NSString **)afterString
```

Parameters

aString

The string that is replacing the characters in *charRange*.

charRange
The range of characters which *aString* is replacing.

beforeString
On return, a pointer to the string with the characters that should be added before *aString*; `nil` if there are no characters to add, if *aString* is `nil`, or if smart insertion and deletion are disabled.

afterString
On return, a pointer to the string with the characters that should be added after *aString*; `nil` if there are no characters to add, if *aString* is `nil`, or if smart insertion and deletion are disabled.

Discussion
As part of its implementation, this method calls `smartInsertAfterStringForString:replacingRange:` (page 86) and `smartInsertBeforeStringForString:replacingRange:` (page 86). To change this method's behavior, override those two methods instead of this one.

`NSTextView` uses this method as necessary. You can also use it in implementing your own methods that insert text. To do so, invoke this method with the proper arguments, then insert *beforeString*, *aString*, and *afterString* in order over *charRange*.

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

See Also
- `smartDeleteRangeForProposedRange:` (page 85)
- `smartInsertDeleteEnabled` (page 87)

Declared In
`NSTextView.h`

spellCheckerDocumentTag

Returns a tag identifying the text view's text as a document for the spell checker server.

```
- (NSInteger)spellCheckerDocumentTag
```

Return Value
A tag identifying the text view's text as a document for the spell checker server.

Discussion
The document tag is obtained by sending a `uniqueSpellDocumentTag` message to the spell server the first time this method is invoked for a particular group of text views. See the `NSSpellChecker`and `NSSpellServer`class specifications for more information on how this tag is used.

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

Declared In
`NSTextView.h`

startSpeaking:

Speaks the selected text, or all text if no selection.

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

Parameters

sender

The control sending the message; can be `nil`.

Availability
Available in Mac OS X v10.1 and later.

See Also
`- stopSpeaking:` (page 89)

Declared In
`NSTextView.h`

stopSpeaking:

Stops the speaking of text.

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

Parameters

sender

The control sending the message; can be `nil`.

Availability
Available in Mac OS X v10.1 and later.

See Also
`- startSpeaking:` (page 88)

Declared In
`NSTextView.h`

textContainer

Returns the receiver's text container.

```
- (NSTextContainer *)textContainer
```

Return Value
The receiver's text container.

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

See Also
`- setTextContainer:` (page 79)

Related Sample Code
LayoutManagerDemo
Sketch-112
TipWrapper

Declared In
NSTextView.h

textContainerInset

Returns the empty space the receiver leaves around its text container.

```
- (NSSize)textContainerInset
```

Return Value
The empty space the receiver leaves around its text container.

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

See Also
- textContainerOrigin (page 90)
- invalidateTextContainerOrigin (page 38)
- setTextContainerInset: (page 80)

Declared In
NSTextView.h

textContainerOrigin

Returns the origin of the receiver's text container.

```
- (NSPoint)textContainerOrigin
```

Return Value
The origin of the receiver's text container, which is calculated from the receiver's bounds rectangle, container inset, and the container's used rect.

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

See Also
- invalidateTextContainerOrigin (page 38)
- textContainerInset (page 90)
- usedRectForTextContainer: (NSLayoutManager)

Related Sample Code
LayoutManagerDemo

Declared In
NSTextView.h

textStorage

Returns the receiver's text storage object.

```
- (NSTextStorage *)textStorage
```

Return Value
The receiver's text storage object.

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

Related Sample Code
BackgroundExporter
NewsReader
TextLinks
VertexPerformanceTest
WebKitPluginWithSimpleGUI

Declared In
`NSTextView.h`

tightenKerning:

Decreases the space between glyphs in the receiver's selection, or for all glyphs if the receiver is a plain text view.

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

Parameters

sender
The control that sent the message; may be `nil`.

Discussion
Kerning values are determined by the point size of the fonts in the selection.

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

See Also
- `loosenKerning:` (page 43)
- `useStandardKerning:` (page 99)
- `turnOffKerning:` (page 94)

Declared In
`NSTextView.h`

toggleAutomaticLinkDetection:

Changes the state of automatic link detection from enabled to disabled and vice versa.

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

Parameters

sender
The control sending the message; may be `nil`.

Discussion
Automatic link detection causes strings representing URLs typed in the view to be automatically made into links to those URLs.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setAutomaticLinkDetectionEnabled:` (page 65)
- `isAutomaticLinkDetectionEnabled` (page 39)
`URLAtIndex:effectiveRange:` (`NSAttributedString`)

Declared In
`NSTextView.h`

toggleAutomaticQuoteSubstitution:

Changes the state of automatic quotation mark substitution from enabled to disabled and vice versa.

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

Parameters
sender
The control sending the message; may be `nil`.

Discussion
Automatic quote substitution causes ASCII quotation marks and apostrophes to be automatically replaced, on a context-dependent basis, with more typographically accurate symbols.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `isAutomaticQuoteSubstitutionEnabled` (page 39)
- `setAutomaticQuoteSubstitutionEnabled:` (page 65)

Declared In
`NSTextView.h`

toggleBaseWritingDirection:

Changes the base writing direction of a paragraph between left-to-right and right-to-left.

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

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSTextView.h`

toggleContinuousSpellChecking:

Toggles whether continuous spell checking is enabled for the receiver.

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

Parameters

sender

The control sending the message; may be `nil`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `isContinuousSpellCheckingEnabled` (page 40)
- `setContinuousSpellCheckingEnabled:` (page 67)

Declared In

`NSTextView.h`

toggleGrammarChecking:

Changes the state of grammar checking from enabled to disabled and vice versa.

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

Parameters

sender

The control sending the message; may be `nil`.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `setGrammarCheckingEnabled:` (page 70)
- `isGrammarCheckingEnabled` (page 41)

Declared In

`NSTextView.h`

toggleSmartInsertDelete:

Changes the state of smart insert and delete from enabled to disabled and vice versa.

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

Parameters

sender

The control sending the message; may be `nil`.

Discussion

Controls whether the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `smartInsertDeleteEnabled` (page 87)
- `setSmartInsertDeleteEnabled:` (page 78)

Declared In
`NSTextView.h`

toggleTraditionalCharacterShape:

Toggles the `NSCharacterShapeAttributeName` attribute at the current selection.

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

Parameters

sender
The control that sent the message; may be `nil`.

Discussion
The `NSCharacterShapeAttributeName` constant is defined in *NSAttributedString Application Kit Additions Reference*.

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

Declared In
`NSTextView.h`

turnOffKerning:

Sets the receiver to use nominal glyph spacing for the glyphs in its selection, or for all glyphs if the receiver is a plain text view.

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

Parameters

sender
The control that sent the message; may be `nil`.

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

See Also
- `useStandardKerning:` (page 99)
- `loosenKerning:` (page 43)
- `tightenKerning:` (page 91)
- `isRichText` (page 41)

Declared In
`NSTextView.h`

turnOffLigatures:

Sets the receiver to use only required ligatures when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.

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

Parameters

sender

The control that sent the message; may be `nil`.

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

See Also
- `useAllLigatures:` (page 97)
- `isRichText` (page 41)
- `useStandardLigatures:` (page 99)

Declared In
`NSTextView.h`

typingAttributes

Returns the current typing attributes.

```
- (NSDictionary *)typingAttributes
```

Return Value
A dictionary of the current typing attributes.

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

See Also
- `setTypingAttributes:` (page 81)

Declared In
`NSTextView.h`

updateDragTypeRegistration

Updates the acceptable drag types of all text views associated with the receiver's layout manager.

```
- (void)updateDragTypeRegistration
```

Discussion
If the receiver is editable and is a rich text view, causes all text views associated with the receiver's layout manager to register their acceptable drag types. If the text view isn't editable or isn't rich text, causes those text views to unregister their dragged types.

Subclasses can override this method to change the conditions for registering and unregistering drag types, whether as a group or individually based on the current state of the text view. They should invoke this method when that state changes to perform the necessary update.

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

See Also
- `acceptableDragTypes` (page 23)
- `registerForDraggedTypes:` (NSView)
- `unregisterDraggedTypes` (NSView)
- `isEditable` (page 40)
- `importsGraphics` (page 35)
- `isRichText` (page 41)

Declared In
`NSTextView.h`

updateFontPanel

Updates the Font panel to contain the font attributes of the selection.

```
- (void)updateFontPanel
```

Discussion
Does nothing if the receiver doesn't use the Font panel. You should never need to invoke this method directly, but you can override it if needed to handle additional font attributes.

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

See Also
- `usesFontPanel` (page 98)

Declared In
`NSTextView.h`

updateInsertionPointStateAndRestartTimer:

Updates the insertion point's location and optionally restarts the blinking cursor timer.

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

Parameters
flag
`YES` to restart the blinking cursor timer, `NO` otherwise.

Discussion
This method is invoked automatically whenever the insertion point needs to be moved; you should never need to invoke it directly, but you can override it to modify insertion point behavior.

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

See Also
- `shouldDrawInsertionPoint` (page 84)

- drawInsertionPointInRect:color:turnedOn: (page 34)

Declared In
NSTextView.h

updateRuler

Updates the ruler view in the receiver's enclosing scroll view to reflect the selection's paragraph and marker attributes.

- (void)updateRuler

Discussion
Does nothing if the ruler isn't visible or if the receiver doesn't use the ruler. You should never need to invoke this method directly, but you can override this method if needed to handle additional ruler attributes.

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

See Also
- usesRuler (page 98)

Declared In
NSTextView.h

useAllLigatures:

Sets the receiver to use all ligatures available for the fonts and languages used when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.

- (void)useAllLigatures:(id)*sender*

Parameters
sender
The control that sent the message; may be nil.

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

See Also
- turnOffLigatures: (page 95)
- useStandardLigatures: (page 99)

Declared In
NSTextView.h

usesFindPanel

Returns whether the receiver allows for a find panel.

- (BOOL)usesFindPanel

Return Value
YES if the receiver allows the use of a find panel, NO otherwise.

Availability
Available in Mac OS X v10.3 and later.

See Also
- setUsesFindPanel: (page 81)

Declared In
NSTextView.h

usesFontPanel

Returns whether the text views sharing the receiver's layout manager use the Font panel.

```
- (BOOL)usesFontPanel
```

Return Value
YES if the text views sharing the receiver's layout manager use the Font panel, NO otherwise.

Discussion
See setUsesFontPanel: (page 82) and rangeForUserCharacterAttributeChange (page 49) for the effect this method has on a text view's behavior.

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

Declared In
NSTextView.h

usesRuler

Returns whether the text views sharing the receiver's layout manager use a ruler.

```
- (BOOL)usesRuler
```

Return Value
YES if the text views sharing the receiver's layout manager use a ruler, NO otherwise.

Discussion
See setUsesRuler: (page 82) and rangeForUserParagraphAttributeChange (page 50) for the effect this has on a text view's behavior. By default, text view objects use the ruler.

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

See Also
- setUsesRuler: (page 82)

Declared In
NSTextView.h

useStandardKerning:

Set the receiver to use pair kerning data for the glyphs in its selection, or for all glyphs if the receiver is a plain text view.

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

Parameters

sender

The control that sent the message; may be `nil`.

Discussion

This data is taken from a font's AFM file

Availability

Available in Mac OS X v10.0 and later.

See Also

- `isRichText` (page 41)
- `loosenKerning:` (page 43)
- `tightenKerning:` (page 91)
- `turnOffKerning:` (page 94)

Declared In

`NSTextView.h`

useStandardLigatures:

Sets the receiver to use the standard ligatures available for the fonts and languages used when setting text, for the glyphs in the selection if the receiver is a rich text view, or for all glyphs if it's a plain text view.

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

Parameters

sender

The control that sent the message; may be `nil`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `turnOffLigatures:` (page 95)
- `useAllLigatures:` (page 97)

Declared In

`NSTextView.h`

validRequestorForSendType:returnType:

Returns `self` if the text view can provide and accept the specified data types, or `nil` if it can't.

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

Parameters

sendType

The type of data requested.

returnType

The type of data that will be returned.

Return Value

`self` if *sendType* specifies a type of data the text view can put on the pasteboard and *returnType* contains a type of data the text view can read from the pasteboard; otherwise `nil`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- validRequestorForSendType:returnType:` (NSResponder)

Declared In

`NSTextView.h`

writablePasteboardTypes

Returns the pasteboard types that can be provided from the current selection.

```
- (NSArray *)writablePasteboardTypes
```

Return Value

An array of strings describing the types that can be written to the pasteboard immediately, or an array with no members if the text view has no text or no selection.

Discussion

Overriders can copy the result from super and add their own new types.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- readablePasteboardTypes` (page 52)

Declared In

`NSTextView.h`

writeSelectionToPasteboard:type:

Writes the current selection to the specified pasteboard using the given type.

```
- (BOOL)writeSelectionToPasteboard:(NSPasteboard *)pboard type:(NSString *)type
```

Parameters

pboard

The pasteboard to write to.

type

The type of data to write.

Return Value
`YES` if the data was successfully written, `NO` otherwise.

Discussion
The complete set of data types being written to *pboard* should be declared before invoking this method.

This method should be invoked only from `writeSelectionToPasteboard:types:` (page 101). You can override this method to add support for writing new types of data to the pasteboard. You should invoke `super`'s implementation of the method to handle any types of data your overridden version does not.

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

See Also
`- readSelectionFromPasteboard:type:` (page 54)

Declared In
`NSTextView.h`

writeSelectionToPasteboard:types:

Writes the current selection to the specified pasteboard under each given type.

```
- (BOOL)writeSelectionToPasteboard:(NSPasteboard *)pboard types:(NSArray *)types
```

Parameters
pboard
The pasteboard to write to.

types
An array of strings describing the types of data to write.

Return Value
`YES` if the data for any single type was successfully written, `NO` otherwise.

Discussion
This method declares the data types on *pboard* and then invokes `writeSelectionToPasteboard:type:` (page 100) or the delegate method `textView:writeCell:atIndex:toPasteboard:type:` (page 112) for each type in the *types* array.

You should not need to override this method. You might need to invoke this method if you are implementing a new type of pasteboard to handle services other than copy/paste or dragging.

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

Declared In
`NSTextView.h`

Delegate Methods

textView:clickedOnCell:inRect:

Sent when the user clicks a cell. (**Deprecated.** Use `textView:clickedOnCell:inRect:atIndex:` (page 102) instead.)

```
- (void)textView:(NSTextView *)aTextView clickedOnCell:(id < NSTextAttachmentCell
    >)attachmentCell inRect:(NSRect)cellFrame
```

Parameters

aTextView

The text view sending the message.

attachmentCell

The cell clicked by the user.

cellFrame

The frame of the clicked cell.

Discussion

This message is only sent if `textView:clickedOnCell:inRect:atIndex:` (page 102) is not implemented. Implement this method in order to track the mouse after a mouse click on a cell.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextView.h`

textView:clickedOnCell:inRect:atIndex:

Sent when the user clicks a cell.

```
- (void)textView:(NSTextView *)aTextView clickedOnCell:(id < NSTextAttachmentCell
    >)cell inRect:(NSRect)cellFrame atIndex:(NSUInteger)charIndex
```

Parameters

aTextView

The text view sending the message.

cell

The cell clicked by the user.

cellFrame

The frame of the clicked cell.

charIndex

The character index of the clicked cell.

Discussion

The delegate can use this message as its cue to perform an action or select the attachment cell's character. *aTextView* is the first text view in a series shared by a layout manager, not necessarily the one that draws *cell*.

The delegate may subsequently receive a `textView:doubleClickedOnCell:inRect:atIndex:` (page 106) message if the user continues to perform a double click.

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

See Also
- `textView:doubleClickedOnCell:inRect:atIndex:` (page 106)

Declared In
`NSTextView.h`

textView:clickedOnLink:

Sent after the user clicks on a link. (**Deprecated.** Use `textView:clickedOnLink:atIndex:` (page 103) instead.)

```
- (BOOL)textView:(NSTextView *)aTextView clickedOnLink:(id)link
```

Parameters

aTextView
The text view sending the message.

link
The link that was clicked.

Discussion
This message is only sent if `textView:clickedOnLink:atIndex:` (page 103) is not implemented.

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

See Also
- `clickedOnLink:atIndex:` (page 28) (NSTextView)
- `textView:clickedOnLink:atIndex:` (page 103)

Declared In
`NSTextView.h`

textView:clickedOnLink:atIndex:

Sent after the user clicks a link.

```
- (BOOL)textView:(NSTextView *)aTextView clickedOnLink:(id)link
    atIndex:(NSUInteger)charIndex
```

Parameters

aTextView
The text view sending the message.

link
The link that was clicked; the value of `NSLinkAttributeName`.

charIndex
The character index where the click occurred, indexed within the text storage.

Return Value
YES if the click was handled; otherwise, NO to allow the next responder to handle it.

Discussion
The delegate can use this method to handle the click on the link. It is invoked by clickedOnLink:atIndex: (page 28).

The *charIndex* parameter is a character index somewhere in the range of the link attribute. If the user actually physically clicked the link, then it should be the character that was originally clicked. In some cases a link may be opened indirectly or programmatically, in which case a character index somewhere in the range of the link attribute is supplied.

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

See Also
- clickedOnLink:atIndex: (page 28) (NSTextView)

Declared In
NSTextView.h

textView:completions:forPartialWordRange:indexOfSelectedItem:

Returns the actual completions for a partial word.

```
- (NSArray *)textView:(NSTextView *)textView completions:(NSArray *)words
    forPartialWordRange:(NSRange)charRange indexOfSelectedItem:(NSInteger *)index
```

Parameters
textView
The text view sending the message.

words
The proposed array of completions.

charRange
The range of characters to be completed.

index
On return, the index of the initially selected completion. The default is 0, and –1 indicates no selection.

Return Value
The actual array of completions that will be presented for the partial word at the given range. Returning nil or a zero-length array suppresses completion.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTextView.h

textView:doCommandBySelector:

Sent to allow the delegate to perform the command for the text view.

```
- (BOOL)textView:(NSTextView *)aTextView doCommandBySelector:(SEL)aSelector
```

Parameters

aTextView

The text view sending the message. This is the first text view in a series shared by a layout manager.

aSelector

The selector.

Return Value

`YES` indicates that the delegate handled the command and the text view will not attempt to perform it; `NO` indicates that the delegate did not handle the command the text view will attempt to perform it.

Discussion

This method is invoked by `NSTextView`'s `doCommandBySelector:` method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextView.h`

textView:doubleClickedOnCell:inRect:

Sent when the user double-clicks a cell. (**Deprecated.** Use `textView:doubleClickedOnCell:inRect:atIndex:` (page 106) instead.)

```
- (void)textView:(NSTextView *)aTextView doubleClickedOnCell:(id <
    NSTextAttachmentCell >)attachmentCell inRect:(NSRect)cellFrame
```

Parameters

aTextView

The text view sending the message.

cell

The cell double-clicked by the user.

cellFrame

The frame of the double-clicked cell.

Discussion

This message is only sent if `textView:doubleClickedOnCell:inRect:atIndex:` (page 106) is not implemented. Implement this method in order to track the mouse after a mouse double-click on a cell.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- textView:doubleClickedOnCell:inRect:atIndex:` (page 106)

Declared In

`NSTextView.h`

textView:doubleClickedOnCell:inRect:atIndex:

Sent when the user double-clicks a cell.

```
- (void)textView:(NSTextView *)aTextView doubleClickedOnCell:(id <
    NSTextAttachmentCell >)cell inRect:(NSRect)cellFrame
    atIndex:(NSUInteger)charIndex
```

Parameters

aTextView

The text view sending the message.

cell

The cell double-clicked by the user.

cellFrame

The frame of the double-clicked cell.

charIndex

The character index of the double-clicked cell.

Discussion

The delegate can use this message as its cue to perform an action, such as opening the file represented by the attachment. *aTextView* is the first text view in a series shared by a layout manager, not necessarily the one that draws *cell*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSTextView.h

textView:draggedCell:inRect:event:

Sent when the user attempts to drag a cell. (**Deprecated.** Use `textView:draggedCell:inRect:event:atIndex:` (page 107) instead.)

```
- (void)textView:(NSTextView *)aTextView draggedCell:(id < NSTextAttachmentCell
    >)cell inRect:(NSRect)aRect event:(NSEvent *)theEvent
```

Parameters

aTextView

The text view sending the message.

cell

The cell being dragged.

aRect

The rectangle from which the cell was dragged.

theEvent

The mouse-down event that preceded the mouse-dragged event.

Discussion

theEvent is the mouse-down event that preceded the mouse-dragged event.

This method has been deprecated in favor of `textView:draggedCell:inRect:event:atIndex:` (page 107).

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

See Also
- `dragImage:at:offset:event:pasteboard:source:slideBack:` (NSView)
- `dragFile:fromRect:slideBack:event:` (NSView)

Declared In
`NSTextView.h`

textView:draggedCell:inRect:event:atIndex:

Sent when the user attempts to drag a cell.

```
- (void)textView:(NSTextView *)aTextView draggedCell:(id < NSTextAttachmentCell
    >)cell inRect:(NSRect)rect event:(NSEvent *)event atIndex:(NSUInteger)charIndex
```

Parameters

aTextView
The text view sending the message.

cell
The cell being dragged.

aRect
The rectangle from which the cell was dragged.

theEvent
The mouse-down event that preceded the mouse-dragged event.

charIndex
The character position where the mouse button was clicked.

Discussion
The delegate can use this message as its cue to initiate a dragging operation.

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

See Also
- `dragImage:at:offset:event:pasteboard:source:slideBack:` (NSView)
- `dragFile:fromRect:slideBack:event:` (NSView)

Declared In
`NSTextView.h`

textView:shouldChangeTextInRange:replacementString:

Sent when a text view needs to determine if text in a specified range should be changed.

```
- (BOOL)textView:(NSTextView *)aTextView
    shouldChangeTextInRange:(NSRange)affectedCharRange replacementString:(NSString
     *)replacementString
```

Parameters

aTextView

The text view sending the message. This is the first text view in a series shared by a layout manager, not necessarily the text view displaying the selected text.

affectedCharRange

The range of characters to be replaced.

replacementString

The characters that will replace the characters in *affectedCharRange*; `nil` if only text attributes are being changed.

Return Value

`YES` to allow the replacement, or `NO` to reject the change.

Discussion

If a delegate implements this method and not its multiple-selection replacement, `textView:shouldChangeTextInRanges:replacementStrings:` (page 108), it is called with an appropriate range and string. If a delegate implements the new method, then this one is ignored.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTextView.h`

textView:shouldChangeTextInRanges:replacementStrings:

Sent when a text view needs to determine if text in an array of specified ranges should be changed.

```
- (BOOL)textView:(NSTextView *)textView shouldChangeTextInRanges:(NSArray
    *)affectedRanges replacementStrings:(NSArray *)replacementStrings
```

Parameters

textView

The text view sending the message. This is the first text view in a series shared by a layout manager, not necessarily the text view displaying the selected text.

affectedRanges

The array of ranges of characters to be replaced. This array must be a non-nil, non-empty array of objects responding to the NSValue `rangeValue` method, and in addition its elements must be sorted, non-overlapping, non-contiguous, and (except for the case of a single range) have non-zero-length.

replacementStrings

The array of strings that will replace the characters in *affectedRanges*, one string for each range; `nil` if only text attributes are being changed.

Return Value

`YES` to allow the replacement, or `NO` to reject the change.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSTextView.h`

textView:shouldChangeTypingAttributes:toAttributes:

Sent when the typing attributes are changed.

```
- (NSDictionary *)textView:(NSTextView *)textView
    shouldChangeTypingAttributes:(NSDictionary *)oldTypingAttributes
    toAttributes:(NSDictionary *)newTypingAttributes
```

Parameters

textView

The text view sending the message.

oldTypingAttributes

The old typing attributes.

newTypingAttributes

The proposed typing attributes.

Return Value
The actual new typing attributes.

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

Declared In
NSTextView.h

textView:shouldSetSpellingState:range:

Sent when the spelling state is changed.

```
- (NSInteger)textView:(NSTextView *)textView
    shouldSetSpellingState:(NSInteger)value
    range:(NSRange)affectedCharRange
```

Parameters

textView

The text view sending the message.

value

The proposed spelling state value to set. Possible values, for the temporary attribute on the layout manager using the key `NSSpellingStateAttributeName`, are:

`NSSpellingStateSpellingFlag` to highlight spelling issues.

`NSSpellingStateGrammarFlag` to highlight grammar issues.

affectedCharRange

The character range over which to set the given spelling state.

Return Value
The actual spelling state to set.

Discussion
Delegate only. Allows delegate to control the setting of spelling and grammar indicators.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setSpellingState:range:` (page 79)

Declared In
`NSTextView.h`

textView:willChangeSelectionFromCharacterRange:toCharacterRange:

Returns the actual range to select.

```
- (NSRange)textView:(NSTextView *)aTextView
    willChangeSelectionFromCharacterRange:(NSRange)oldSelectedCharRange
    toCharacterRange:(NSRange)newSelectedCharRange
```

Parameters

aTextView
The text view sending the message. This is the first text view in a series shared by a layout manager, not necessarily the text view displaying the selected text.

oldSelectedCharRange
The original range of the selection.

newSelectedCharRange
The proposed character range for the new selection.

Return Value
The actual character range for the new selection.

Discussion
This method is invoked before a text view finishes changing the selection—that is, when the last argument to a `setSelectedRange:affinity:stillSelecting:` (page 75) message is `NO`.

Special Considerations

In Mac OS X version 10.4 and later, if a delegate implements this delegate method and not its multiple-selection replacement, `textView:willChangeSelectionFromCharacterRanges:toCharacterRanges:` (page 110), then multiple selection is effectively disallowed; attempts to set the selected ranges call the old delegate method with the first subrange, and afterwards only a single selected range is set.

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

See Also
- `textView:willChangeSelectionFromCharacterRanges:toCharacterRanges:` (page 110)

Declared In
`NSTextView.h`

textView:willChangeSelectionFromCharacterRanges:toCharacterRanges:

Returns the actual character ranges to select.

```
- (NSArray *)textView:(NSTextView *)aTextView
    willChangeSelectionFromCharacterRanges:(NSArray *)oldSelectedCharRanges
    toCharacterRanges:(NSArray *)newSelectedCharRanges
```

Parameters

aTextView

The text view sending the message. This is the first text view in a series shared by a layout manager, not necessarily the text view displaying the selected text.

oldSelectedCharRanges

An array containing the original ranges of the selection. This must be a non-nil, non-empty array of objects responding to the NSValue rangeValue method, and in addition its elements must be sorted, non-overlapping, non-contiguous, and (except for the case of a single range) have non-zero-length.

newSelectedCharRanges

An array containing the proposed character ranges for the new selection. This must be a non-nil, non-empty array of objects responding to the NSValue rangeValue method, and in addition its elements must be sorted, non-overlapping, non-contiguous, and (except for the case of a single range) have non-zero-length.

Return Value

An array containing the actual character ranges for the new selection.

Discussion

Invoked before an NSTextView finishes changing the selection—that is, when the last argument to a setSelectedRange:affinity:stillSelecting: (page 75) or setSelectedRanges:affinity:stillSelecting: (page 77) message is NO.

If a delegate implements both this method and textView:willChangeSelectionFromCharacterRange:toCharacterRange: (page 110), then the latter is ignored.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSTextView.h

textView:willDisplayToolTip:forCharacterAtIndex:

Returns the actual tooltip to display.

```
- (NSString *)textView:(NSTextView *)textView willDisplayToolTip:(NSString *)tooltip
    forCharacterAtIndex:(NSUInteger)characterIndex
```

Parameters

textView

The text view sending the message.

tooltip

The proposed tooltip to display.

characterIndex

The location in *textView*.

Return Value

The actual tooltip to display, or nil to suppress display of the tooltip.

Discussion

The tooltip string is the value of the NSToolTipAttributeName attribute at *characterIndex*.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSTextView.h

textView:writablePasteboardTypesForCell:atIndex:

Returns the writable pasteboard types for a given cell.

```
- (NSArray *)textView:(NSTextView *)aTextView writablePasteboardTypesForCell:(id
    < NSTextAttachmentCell >)cell atIndex:(NSUInteger)charIndex
```

Parameters

aTextView
The text view sending the message.

cell
The cell in question.

charIndex
The character index in the text view that was clicked.

Return Value
An array of types that can be written to the pasteboard for *cell*.

Discussion
This method is invoked after the user clicks *cell* at the specified *charIndex* location in *aTextView*. If the `textView:draggedCell:inRect:event:atIndex:` (page 107) is not used, this method and `textView:writeCell:atIndex:toPasteboard:type:` (page 112) allow *aTextView* to take care of attachment dragging and pasting, with the delegate responsible only for writing the attachment to the pasteboard.

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

Declared In
NSTextView.h

textView:writeCell:atIndex:toPasteboard:type:

Returns whether data of the specified type for the given cell could be written to the specified pasteboard.

```
- (BOOL)textView:(NSTextView *)aTextView writeCell:(id < NSTextAttachmentCell >)cell
    atIndex:(NSUInteger)charIndex toPasteboard:(NSPasteboard *)pboard type:(NSString
    *)type
```

Parameters

aTextView
The text view sending the message.

cell
The cell whose contents should be written to the pasteboard.

charIndex
The index at which the cell was accessed.

pboard
The pasteboard to which the cell's contents should be written.

type
The type of data that should be written.

Return Value
YES if the write succeeded, NO otherwise.

Discussion
The receiver should attempt to write the *cell* to *pboard* with the given *type*, and return success or failure.

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

Declared In
NSTextView.h

textViewDidChangeSelection:

Sent when the selection changes in the text view.

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

Parameters
aNotification
The notification.

Discussion
The name of *aNotification* is NSTextViewDidChangeSelectionNotification (page 119).

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

Declared In
NSTextView.h

textViewDidChangeTypingAttributes:

Sent when a text view's typing attributes change.

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

Parameters
aNotification
The notification object. The default name of *aNotification* is NSTextViewDidChangeTypingAttributesNotification (page 120).

Discussion
This method allows the delegate to modify the notification sent when the typing attributes of a text view change.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSTextView.h`

undoManagerForTextView:

Returns the undo manager for the specified text view.

```
- (NSUndoManager *)undoManagerForTextView:(NSTextView *)aTextView
```

Parameters

aTextView

The text view whose undo manager should be returned.

Return Value
The undo manager for *aTextView*.

Discussion
This method provides the flexibility to return a custom undo manager for the text view. Although `NSTextView` implements undo and redo for changes to text, applications may need a custom undo manager to handle interactions between changes to text and changes to other items in the application.

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

Declared In
`NSTextView.h`

Constants

NSSelectionGranularity

These constants specify how much the text view extends the selection when the user drags the mouse. They're used by `selectionGranularity` (page 61), `setSelectionGranularity:` (page 78), and `selectionRangeForProposedRange:granularity:` (page 61):

```
typedef enum _NSSelectionGranularity {
    NSSelectByCharacter = 0,
    NSSelectByWord = 1,
    NSSelectByParagraph = 2
} NSSelectionGranularity;
```

Constants

`NSSelectByCharacter`

Extends the selection character by character.

Available in Mac OS X v10.0 and later.

Declared in `NSTextView.h`.

`NSSelectByWord`
Extends the selection word by word.

Available in Mac OS X v10.0 and later.

Declared in `NSTextView.h`.

`NSSelectByParagraph`
Extends the selection paragraph by paragraph.

Available in Mac OS X v10.0 and later.

Declared in `NSTextView.h`.

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

Declared In
`NSTextView.h`

NSSelectionAffinity

These constants specify the preferred direction of selection. They're used by `selectionAffinity` (page 61) and `setSelectedRange:affinity:stillSelecting:` (page 75).

```
typedef enum _NSSelectionAffinity {
    NSSelectionAffinityUpstream = 0,
    NSSelectionAffinityDownstream = 1
} NSSelectionAffinity;
```

Constants
`NSSelectionAffinityUpstream`
The selection is moving toward the top of the document.

Available in Mac OS X v10.0 and later.

Declared in `NSTextView.h`.

`NSSelectionAffinityDownstream`
The selection is moving toward the bottom of the document.

Available in Mac OS X v10.0 and later.

Declared in `NSTextView.h`.

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

Declared In
`NSTextView.h`

NSFindPanelAction

These constants define the tags for `performFindPanelAction:` (page 47).

```
typedef enum {
    NSFindPanelActionShowFindPanel = 1,
    NSFindPanelActionNext = 2,
    NSFindPanelActionPrevious = 3,
    NSFindPanelActionReplaceAll = 4,
    NSFindPanelActionReplace = 5,
    NSFindPanelActionReplaceAndFind = 6,
    NSFindPanelActionSetFindString = 7,
    NSFindPanelActionReplaceAllInSelection = 8
} NSFindPanelAction;
```

Constants

`NSFindPanelActionShowFindPanel`
Displays the find panel.
Available in Mac OS X v10.3 and later.
Declared in `NSTextView.h`.

`NSFindPanelActionNext`
Finds the next instance of the queried text.
Available in Mac OS X v10.3 and later.
Declared in `NSTextView.h`.

`NSFindPanelActionPrevious`
Finds the previous instance of the queried text.
Available in Mac OS X v10.3 and later.
Declared in `NSTextView.h`.

`NSFindPanelActionReplaceAll`
Replaces all query instances within the text view.
Available in Mac OS X v10.3 and later.
Declared in `NSTextView.h`.

`NSFindPanelActionReplace`
Replaces a single query instance within the text view.
Available in Mac OS X v10.3 and later.
Declared in `NSTextView.h`.

`NSFindPanelActionReplaceAndFind`
Replaces a single query instance and finds the next.
Available in Mac OS X v10.3 and later.
Declared in `NSTextView.h`.

`NSFindPanelActionSetFindString`
Sets the query string to the current selection.
Available in Mac OS X v10.3 and later.
Declared in `NSTextView.h`.

`NSFindPanelActionReplaceAllInSelection`
Replaces all query instances within the selection.
Available in Mac OS X v10.3 and later.
Declared in `NSTextView.h`.

`NSFindPanelActionSelectAll`
Selects all query instances in the text view.

Available in Mac OS X v10.4 and later.

Declared in `NSTextView.h`.

`NSFindPanelActionSelectAllInSelection`
Selects all query instances within the selection.

Available in Mac OS X v10.4 and later.

Declared in `NSTextView.h`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSTextView.h`

Input Sources Locale Identifiers

Locale identifiers represent the input sources available.

```
NSString *NSAllRomanInputSourcesLocaleIdentifier
```

Constants
`NSAllRomanInputSourcesLocaleIdentifier`
A meta-locale identifier representing the set of Roman input sources available. You can pass `[NSArray arrayWithObject: NSAllRomanInputSourcesLocaleIdentifier]` to the `setAllowedInputSourceLocales:` (page 63) method to restrict allowed input sources to Roman only.

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

Declared In
`NSTextView.h`

Find Panel Search Metadata

In addition to communicating search strings via the find pasteboard, the standard Find panel for `NSTextView` also communicates search option metadata, including case sensitivity and substring matching options. This metadata is stored in a property list as the `NSFindPanelSearchOptionsPboardType` (page 118) value on the global find pasteboard. As such, third party applications may store additional keys in this property list to communicate additional metadata as desired to support the various search options common to many third-party applications' Find panels.

```
NSString *NSFindPanelSearchOptionsPboardType
NSString *NSFindPanelCaseInsensitiveSearch
NSString *NSFindPanelSubstringMatch
```

Constants

`NSFindPanelSearchOptionsPboardType`

Type for `NSFindPanel` metadata property list. Used with the `NSPasteBoard` method `propertyListForType:`.

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

`NSFindPanelCaseInsensitiveSearch`

Boolean value specifying whether the search is case-insensitive. `YES` specifies a case-insensitive search; otherwise, `NO`.

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

`NSFindPanelSubstringMatch`

`NSNumber` object containing one of the values defined in "NSFindPanelSubstringMatchType" (page 118).

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

Declared In

`NSTextView.h`

NSFindPanelSubstringMatchType

The type of substring matching used by the Find panel.

```
enum {
    NSFindPanelSubstringMatchTypeContains = 0,
    NSFindPanelSubstringMatchTypeStartsWith = 1,
    NSFindPanelSubstringMatchTypeFullWord = 2,
    NSFindPanelSubstringMatchTypeEndsWith = 3
};
typedef NSUInteger NSFindPanelSubstringMatchType;
```

Constants

`NSFindPanelSubstringMatchTypeContains`

Finds a word containing the search string.

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

`NSFindPanelSubstringMatchTypeStartsWith`

Finds a word starting with the search string.

Available in Mac OS X v10.5 and later.

Declared in `NSTextView.h`.

`NSFindPanelSubstringMatchTypeFullWord`
: Finds a word exactly matching the search string.

  Available in Mac OS X v10.5 and later.

  Declared in `NSTextView.h`.

`NSFindPanelSubstringMatchTypeEndsWith`
: Finds a word ending with the search string.

  Available in Mac OS X v10.5 and later.

  Declared in `NSTextView.h`.

Declared In
`NSTextView.h`

Notifications

`NSTextView` posts the following notifications as well as those declared by its superclasses, particularly `NSText`. See the "Notifications" section in the `NSText` class specification for those other notifications.

NSTextViewDidChangeSelectionNotification

Posted when the selected range of characters changes.

`NSTextView` posts this notification whenever `setSelectedRange:affinity:stillSelecting:` (page 75) is invoked, either directly or through the many methods (`mouseDown:`, `selectAll:`, and so on) that invoke it indirectly. When the user is selecting text, this notification is posted only once, at the end of the selection operation. The text view's delegate receives a `textViewDidChangeSelection:` (page 113) message when this notification is posted.

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

Key	Value
`@"NSOldSelectedCharacterRange"`	An `NSValue` object containing an `NSRange` structure with the originally selected range.

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

Declared In
`NSTextView.h`

NSTextViewWillChangeNotifyingTextViewNotification

Posted when a new text view is established as the text view that sends notifications.

This notification allows observers to reregister themselves for the new text view. Methods such as `removeTextContainerAtIndex:`, `textContainerChangedTextView:`, and `insertTextContainer:atIndex:` cause this notification to be posted.

The notification object is the old notifying text view, or `nil`. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSOldNotifyingTextView"`	The old `NSTextView`, if one exists, otherwise `nil`.
`@"NSNewNotifyingTextView"`	The new `NSTextView`, if one exists, otherwise `nil`.

There's no delegate method associated with this notification. The text-handling system ensures that when a new text view replaces an old one as the notifying text view, the existing delegate becomes the delegate of the new text view, and the delegate is registered to receive text view notifications from the new notifying text view. All other observers are responsible for registering themselves on receiving this notification.

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

See Also
`- removeObserver:` (`NSNotificationCenter`)
`- addObserver:selector:name:object:` (`NSNotificationCenter`)

Declared In
`NSTextView.h`

NSTextViewDidChangeTypingAttributesNotification

Posted when there is a change in the typing attributes within a text view. This notification is posted, via the `textViewDidChangeTypingAttributes:` (page 113) delegate method, whether or not text has changed as a result of the attribute change.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSTextView.h`

Document Revision History

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

Date	Notes
2007-04-27	Documented methods and constants added in Mac OS X v10.5. Removed descriptions of deprecated methods no longer defined in the header file.
	Added cross-reference to key bindings article in discussion of `complete:` (page 29). Clarified parameter of `insertText:` (page 38).
	Revised description of `setTypingAttributes:` (page 81) method. Fixed external cross reference in `didChangeText` (page 31) method description. Revised descriptions of `shouldChangeTextInRange:replacementString:` (page 83) and `shouldChangeTextInRanges:replacementStrings:` (page 84) methods.
	Revised descriptions of `linkTextAttributes` (page 43) and `setLinkTextAttributes:` (page 72).
	Amplified description of `outline:` (page 46).
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index

L

M

N

O

P

R

S

T

U

V

W