Class

NSTextView

A view that draws text and handles user interactions with that text.

macOS

```
@MainActor
class NSTextView
```

Mentioned in

- Customizing Writing Tools behavior for AppKit views
- Supporting Writing Tools via the pasteboard
- Adding Writing Tools support to a custom AppKit view
- Adopting the system text cursor in custom text views

Overview

The `NSTextView` class is the front-end class to the AppKit text system. The class draws the text managed by the back-end components and handles user events to select and modify its text, in addition to supporting rich text, attachments, input management, and key binding, and marked text attributes.

> Note
>
> If you need only to implement a simple editable text field, see `NSTextField`.

`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 the

NSText class — which declares the most general Cocoa interface to the text system — NSTextView adds major features beyond the capabilities of NSText. You can also do more powerful and more creative text manipulation (such as displaying text in a circle) using NSTextStorage, NSTextLayoutManager, NSTextContainer, and related classes.

You're more likely to use the NSTextView class than NSText. It's also important to remember that NSTextView conforms to a large number of protocols, the methods of which are available to instances of the NSTextView class.

NSTextView communicates with its delegate through methods declared both by the NSTextViewDelegate and by its superclass's protocol, NSTextDelegate. All delegation messages come from the first text view.

In macOS 12 and later, if you explicitly call the layoutManager property on a text view or text container, the framework reverts to a compatibility mode that uses NSLayoutManager. The text view also switches to this compatibility mode when it encounters text content that's not yet supported, such as NSTextTable.

About Delegate Methods

The NSTextView class communicates with its delegate through methods declared both by the NSTextViewDelegate and by its superclass's protocol, NSTextDelegate. All delegation messages come from the first text view.

Becoming the first responder

When the system invokes becomeFirstResponder() on a text view, if the previous first responder was not a text view on the same layout manager as the receiving text view, the receiving text view draws the selection and updates the insertion point if necessary.

To make a text view the first responder, call the containing window's makeFirstResponder(_:) method. Never invoke a text view's becomeFirstResponder() method directly.

Resigning as first responder

When the system invokes resignFirstResponder() on a text view, if the object that will become the new first responder is a text view attached to the same layout manager as the receiver, the receiving text view returns true with no further action. Otherwise, it sends a textShouldEndEditing(_:) message to its delegate (if any). If the delegate returns false, the text view returns false. If the delegate returns true, the text view hides the selection highlighting and posts an didEndEditingNotification to the default notification center and then returns true.

Topics

Creating a text view

`init(frame: NSRect, textContainer: NSTextContainer?)`

Initializes a text view.

`init(frame: NSRect)`

Initializes a text view.

`convenience init(usingTextLayoutManager: Bool)`

`init?(coder: NSCoder)`

Initializes a text view with data in an unarchiver.

Managing the text view’s content

`var delegate: (any NSTextViewDelegate)?`

The delegate for all text views sharing the receiver’s layout manager.

`protocol NSTextViewDelegate`

A set of optional methods that text view delegates can use to manage selection, set text attributes, work with the spell checker, and more.

Registering services information

`class func registerForServices()`

Registers send and return types for the Services facility.

Accessing text system objects

`class var stronglyReferencesTextStorage: Bool`

`class func fieldEditor() -> Self`

`var textContainer: NSTextContainer?`

The receiver’s text container.

`func replaceTextContainer(NSTextContainer)`

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.

`var textContainerInset: NSSize`

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

`var textContainerOrigin: NSPoint`

The origin of the receiver's text container.

`func invalidateTextContainerOrigin()`

Invalidates the calculated origin of the text container.

`var textLayoutManager: NSTextLayoutManager?`

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

`var layoutManager: NSLayoutManager?`

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

`var textContentStorage: NSTextContentStorage?`

The receiver's text storage object.

`var textStorage: NSTextStorage?`

The receiver's text storage object.

Setting graphics attributes

`var backgroundColor: NSColor`

The receiver's background color.

`var drawsBackground: Bool`

A Boolean value that indicates whether the receiver draws its background.

`var allowsDocumentBackgroundColorChange: Bool`

A Boolean value that indicates whether the receiver allows its background color to change.

`func changeDocumentBackgroundColor(Any?)`

An action method used to set the background color.

Controlling text display

`func setNeedsDisplay(NSRect, avoidAdditionalLayout: Bool)`

Marks the receiver as requiring display.

`var shouldDrawInsertionPoint: Bool`

A Boolean value that determines whether the receiver should draw its insertion point.

`func drawInsertionPoint(in: NSRect, color: NSColor, turnedOn: Bool)`

Draws or erases the insertion point.

`func drawBackground(in: NSRect)`

Draws the background of the text view.

`func setConstrainedFrameSize(NSSize)`

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

`func cleanUpAfterDragOperation()`

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

`func showFindIndicator(for: NSRange)`

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

`class func scrollableDocumentContentTextView() -> NSScrollView`

`class func scrollablePlainDocumentContentTextView() -> NSScrollView`

`class func scrollableTextView() -> NSScrollView`

Inserting text

`var allowedInputSourceLocales: [String]?`

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

~~`func insertText(Any)`~~

Inserts a`String` into the receiver's text at the insertion point if there is one, otherwise replacing the selection.

Deprecated

Setting behavioral attributes

`var allowsUndo: Bool`

A Boolean value that indicates whether the receiver allows undo.

`var isEditable: Bool`

A Boolean value that controls whether the text views sharing the receiver's layout manager allow the user to edit text.

`var isSelectable: Bool`

A Boolean value that controls whether the text views sharing the receiver's layout manager allow the user to select text.

`var isFieldEditor: Bool`

A Boolean value that controls whether the text views sharing the receiver's layout manager behave as field editors.

`var isRichText: Bool`

A Boolean value that controls whether the text views sharing the receiver's layout manager allow the user to apply attributes to specific ranges of text.

`var importsGraphics: Bool`

A Boolean value that controls whether the text views sharing the receiver's layout manager allow the user to import files by dragging.

`func setBaseWritingDirection(NSWritingDirection, range: NSRange)`

Sets the base writing direction of a range of text.

`var defaultParagraphStyle: NSParagraphStyle?`

The receiver's default paragraph style.

`func outline(Any?)`

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

`var allowsImageEditing: Bool`

Indicates whether image attachments should permit editing of their images.

`var isAutomaticQuoteSubstitutionEnabled: Bool`

A Boolean value that enables and disables automatic quotation mark substitution.

`func toggleAutomaticQuoteSubstitution(Any?)`

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

`var isAutomaticLinkDetectionEnabled: Bool`

A Boolean value that enables or disables automatic link detection.

`func toggleAutomaticLinkDetection(Any?)`

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

`var displaysLinkToolTips: Bool`

A Boolean value that indicates whether the text view automatically supplies the destination of a link as a tooltip for text that has a link attribute.

`var isAutomaticTextCompletionEnabled: Bool`

A Boolean value that indicates whether the text view supplies autocompletion suggestions as the user types.

`func toggleAutomaticTextCompletion(Any?)`

`var usesAdaptiveColorMappingForDarkAppearance: Bool`

A Boolean value that indicates whether the framework should use adaptive color mapping for dark appearance.

`var usesRolloverButtonForSelection: Bool`

Using text formatting controls

`var usesRuler: Bool`

A Boolean value that controls whether the text views sharing the receiver's layout manager use a ruler.

`var isRulerVisible: Bool`

A Boolean value that controls whether the scroll view enclosing text views sharing the receiver's layout manager displays the ruler.

`var usesInspectorBar: Bool`

A Boolean value that indicates whether this text view uses the inspector bar.

Managing the selection

`var selectedRanges: [NSValue]`

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

`func setSelectedRange(NSRange)`

Selects the specified range of characters in response to user action.

`func setSelectedRange(NSRange, affinity: NSSelectionAffinity, still Selecting: Bool)`

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

```
func setSelectedRanges([NSValue], affinity: NSSelectionAffinity, still
Selecting: Bool)
```

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

```
var selectionAffinity: NSSelectionAffinity
```

The preferred direction of selection.

```
var selectionGranularity: NSSelectionGranularity
```

The selection granularity for subsequent extension of a selection.

```
var insertionPointColor: NSColor!
```

The color of the insertion point.

```
func updateInsertionPointStateAndRestartTimer(Bool)
```

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

```
var selectedTextAttributes: [NSAttributedString.Key : Any]
```

The attributes used to indicate the selection.

```
var markedTextAttributes: [NSAttributedString.Key : Any]?
```

The attributes used to draw marked text.

```
var linkTextAttributes: [NSAttributedString.Key : Any]?
```

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

```
func characterIndexForInsertion(at: NSPoint) -> Int
```

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

```
func updateCandidates()
```

Managing the pasteboard

```
func preferredPasteboardType(from: [NSPasteboard.PasteboardType],
restrictedToTypesFrom: [NSPasteboard.PasteboardType]?) -> NSPasteboard.
PasteboardType?
```

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

```
func readSelection(from: NSPasteboard) -> Bool
```

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

`func readSelection(from: NSPasteboard, type: NSPasteboard.PasteboardType) -> Bool`

Reads data of the given type from the specified pasteboard.

`var readablePasteboardTypes: [NSPasteboard.PasteboardType]`

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

`var writablePasteboardTypes: [NSPasteboard.PasteboardType]`

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

`func writeSelection(to: NSPasteboard, type: NSPasteboard.PasteboardType) -> Bool`

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

`func writeSelection(to: NSPasteboard, types: [NSPasteboard.PasteboardType]) -> Bool`

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

`func validRequestor(forSendType: NSPasteboard.PasteboardType?, returnType: NSPasteboard.PasteboardType?) -> Any?`

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

Setting text attributes

`func alignJustified(Any?)`

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

`func changeAttributes(Any?)`

Changes the attributes of the current selection.

`func changeColor(Any?)`

Sets the color of the selected text.

`func setAlignment(NSTextAlignment, range: NSRange)`

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

`var typingAttributes: [NSAttributedString.Key : Any]`

The receiver's typing attributes.

`func useStandardKerning(Any?)`

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.

`func lowerBaseline(Any?)`

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

`func raiseBaseline(Any?)`

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

`func turnOffKerning(Any?)`

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.

`func loosenKerning(Any?)`

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

`func tightenKerning(Any?)`

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

`func useStandardLigatures(Any?)`

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.

`func turnOffLigatures(Any?)`

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.

`func useAllLigatures(Any?)`

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.

~~`func toggleTraditionalCharacterShape(Any?)`~~

Toggles the `NSCharacterShapeAttributeName` attribute at the current selection.

Deprecated

Clicking and pasting

`func clicked(onLink: Any, at: Int)`

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.

`func pasteAsPlainText(Any?)`

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

`func pasteAsRichText(Any?)`

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

Supporting undo

`func breakUndoCoalescing()`

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

`var isCoalescingUndo: Bool`

A Boolean value that indicates whether undo coalescing is in progress.

Customizing subclass behaviors

`func updateFontPanel()`

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

`func updateRuler()`

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

`var acceptableDragTypes: [NSPasteboard.PasteboardType]`

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

`func updateDragTypeRegistration()`

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

`func selectionRange(forProposedRange: NSRange, granularity: NSSelectionGranularity) -> NSRange`

Returns an adjusted selected range based on the selection granularity.

`var rangeForUserCharacterAttributeChange: NSRange`

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

```
var rangesForUserCharacterAttributeChange: [NSValue]?
```

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

```
var rangeForUserParagraphAttributeChange: NSRange
```

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

```
var rangesForUserParagraphAttributeChange: [NSValue]?
```

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

```
var rangeForUserTextChange: NSRange
```

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

```
var rangesForUserTextChange: [NSValue]?
```

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

```
func shouldChangeText(in: NSRange, replacementString: String?) -> Bool
```

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.

```
func shouldChangeText(inRanges: [NSValue], replacementStrings: [String
]?) -> Bool
```

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.

```
func didChangeText()
```

Sends out necessary notifications when a text change completes.

```
var smartInsertDeleteEnabled: Bool
```

A Boolean value that controls whether the receiver inserts or deletes space around selected words so as to preserve proper spacing and punctuation.

```
func smartDeleteRange(forProposedRange: NSRange) -> NSRange
```

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.

`func smartInsert(afterStringFor: String, replacing: NSRange) -> String?`

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.

`func smartInsert(beforeStringFor: String, replacing: NSRange) -> String?`

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.

`func smartInsert(for: String, replacing: NSRange, before: AutoreleasingUnsafeMutablePointer<NSString?>?, after: AutoreleasingUnsafeMutablePointer<NSString?>?)`

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.

`func toggleSmartInsertDelete(Any?)`

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

Working with the spelling checker

`var isContinuousSpellCheckingEnabled: Bool`

A Boolean value that indicates whether the receiver has continuous spell checking enabled.

`var spellCheckerDocumentTag: Int`

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

`func toggleContinuousSpellChecking(Any?)`

Toggles whether continuous spell checking is enabled for the receiver.

`var isGrammarCheckingEnabled: Bool`

Enables and disables grammar checking.

`func toggleGrammarChecking(Any?)`

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

`func setSpellingState(Int, range: NSRange)`

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

Working with the sharing service picker

```
func orderFrontSharingServicePicker(Any?)
```

Creates and displays a new instance of the sharing service picker.

Supporting the ruler view

```
func rulerView(NSRulerView, didMove: NSRulerMarker)
```

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

```
func rulerView(NSRulerView, willMove: NSRulerMarker, toLocation:
CGFloat) -> CGFloat
```

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

```
func rulerView(NSRulerView, shouldMove: NSRulerMarker) -> Bool
```

Returns whether the marker should be moved.

```
func rulerView(NSRulerView, didRemove: NSRulerMarker)
```

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

```
func rulerView(NSRulerView, shouldRemove: NSRulerMarker) -> Bool
```

Returns whether the marker should be removed.

```
func rulerView(NSRulerView, didAdd: NSRulerMarker)
```

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

```
func rulerView(NSRulerView, shouldAdd: NSRulerMarker) -> Bool
```

Returns whether a new marker can be added.

```
func rulerView(NSRulerView, willAdd: NSRulerMarker, atLocation: CGFloat
) -> CGFloat
```

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

```
func rulerView(NSRulerView, handleMouseDownWith: NSEvent)
```

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

Dragging

```
func dragImageForSelection(with: NSEvent, origin: NSPointPointer?) ->
NSImage?
```

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

```
func dragOperation(for: any NSDraggingInfo, type: NSPasteboard.
PasteboardType) -> NSDragOperation
```

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

```
func dragSelection(with: NSEvent, offset: NSSize, slideBack: Bool) ->
Bool
```

Begins dragging the current selected text range.

```
var acceptsGlyphInfo: Bool
```

A Boolean value that indicates whether the receiver accepts the glyph info attribute.

Speaking text

```
func startSpeaking(Any?)
```

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

```
func stopSpeaking(Any?)
```

Stops the speaking of text.

Working with panels

```
var usesFontPanel: Bool
```

A Boolean value that controls whether the text views sharing the receiver’s layout manager use the Font panel and Font menu.

```
var usesFindPanel: Bool
```

A Boolean value that indicates whether the receiver allows for a find panel.

```
func performFindPanelAction(Any?)
```

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

```
func orderFrontLinkPanel(Any?)
```

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

```
func orderFrontListPanel(Any?)
```

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

```
func orderFrontSpacingPanel(Any?)
```

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

`func orderFrontTablePanel(Any?)`

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

`func orderFrontSubstitutionsPanel(Any?)`

Brings forward a panel allowing the user to specify string substitutions in the text view.

Performing text completion

`func complete(Any?)`

Invokes completion in a text view.

`func completions(forPartialWordRange: NSRange, indexOfSelectedItem: UnsafeMutablePointer<Int>) -> [String]?`

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

`func insertCompletion(String, forPartialWordRange: NSRange, movement: Int, isFinal: Bool)`

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

`var rangeForUserCompletion: NSRange`

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

Checking and substituting text

`func checkTextInDocument(Any?)`

Performs the default text checking on the entire document.

`func checkTextInSelection(Any?)`

Performs the default text checking on the current selection.

`func checkText(in: NSRange, types: NSTextCheckingTypes, options: [NSSpellChecker.OptionKey : Any])`

Check and replace the text in the range using the specified checking types and options.

`func handleTextCheckingResults([NSTextCheckingResult], forRange: NSRange, types: NSTextCheckingTypes, options: [NSSpellChecker.OptionKey : Any], orthography: NSOrthography, wordCount: Int)`

Handles the text checking results returned by the text view

`var enabledTextCheckingTypes: NSTextCheckingTypes`

The default text checking types.

`var isAutomaticDashSubstitutionEnabled: Bool`

A Boolean value that indicates whether automatic dash substitution is enabled.

`func toggleAutomaticDashSubstitution(Any?)`

Toggles the state of the automatic dash substitution.

`var isAutomaticDataDetectionEnabled: Bool`

A Boolean value that indicates whether automatic data detection is enabled.

`func toggleAutomaticDataDetection(Any?)`

Toggles the state of the automatic data detection.

`var isAutomaticSpellingCorrectionEnabled: Bool`

A Boolean value that indicates whether automatic spelling correction is enabled.

`func toggleAutomaticSpellingCorrection(Any?)`

Toggles the state of the automatic spelling correction.

`var isAutomaticTextReplacementEnabled: Bool`

A Boolean value that indicates whether automatic text replacement is enabled.

`func toggleAutomaticTextReplacement(Any?)`

Toggles the state of the automatic text replacement.

```
func performValidatedReplacement(in: NSRange, with: NSAttributedString)
-> Bool
```

Replaces text in the range you specify with the attributed string you provide.

Getting the writing tools status

`var isWritingToolsActive: Bool`

Supporting QuickLook

`func updateQuickLookPreviewPanel()`

Notifies the QuickLook panel that an update may be required.

`func toggleQuickLookPreviewPanel(Any?)`

An action message that toggles the visibility state of the Quick Look preview panel.

`func quickLookPreviewableItems(inRanges: [NSValue]) -> [any QLPreviewItem]`

Returns an array of URLs for items that can be displayed by QuickLook in the specified ranges.

Changing layout orientation

`func changeLayoutOrientation(Any?)`

An action method that sets the layout orientation of the text.

`func setLayoutOrientation(NSLayoutManager.TextLayoutOrientation)`

Changes the receiver's layout orientation and invalidates the contents.

Using the Find Bar

`var usesFindBar: Bool`

A Boolean value that indicates whether to use the find bar for this text view.

`var isIncrementalSearchingEnabled: Bool`

A Boolean value that indicates whether incremental searching is enabled.

Constants

`enum NSSelectionGranularity`

These constants specify how much the text view extends the selection when the user drags the mouse. They're used by `selectionGranularity`, and `selectionRange(forProposedRange:granularity:)`:

`enum NSSelectionAffinity`

These constants specify the preferred direction of selection. They're used by `selectionAffinity` and `setSelectedRange(_:affinity:stillSelecting:)`.

`enum NSFindPanelAction`

These constants define the tags for `performFindPanelAction(_:)`.

Input Sources Locale Identifiers

Locale identifiers represent the input sources available.

☰ 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 `findPanelSearchOptions` 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.

`enum NSFindPanelSubstringMatchType`

The type of substring matching used by the Find panel.

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.

`class let didChangeSelectionNotification: NSNotification.Name`

Posted when the selected range of characters changes.

`class let willChangeNotifyingTextViewNotification: NSNotification.Name`

Posted when a new text view is established as the text view that sends notifications.

`class let didChangeTypingAttributesNotification: NSNotification.Name`

Posted when there is a change in the typing attributes within a text view.

`class let didSwitchToNSLayoutManagerNotification: NSNotification.Name`

Posted by the framework after switching to using the compatibility mode layout manager.

`class let willSwitchToNSLayoutManagerNotification: NSNotification.Name`

Posted by the framework before switching to the compatibility mode layout manager.

Interacting with the Touch Bar

`var allowsCharacterPickerTouchBarItem: Bool`

`var candidateListTouchBarItem: NSCandidateListTouchBarItem<AnyObject>?`

`func updateTextTouchBarItems()`

`func updateTouchBarItemIdentifiers()`

Instance Properties

`var allowedWritingToolsResultOptions: NSWritingToolsResultOptions`

`var inlinePredictionType: NSTextInputTraitType`

`var mathExpressionCompletionType: NSTextInputTraitType`

`var textHighlightAttributes: [NSAttributedString.Key : Any]`

************************ Text Highlight support ************************

`var writingToolsBehavior: NSWritingToolsBehavior`

Instance Methods

`func drawTextHighlightBackground(for: NSTextRange, origin: NSPoint)`

`func highlight(Any?)`

An action for toggling `NSTextHighlightStyleAttributeName` in the receiver's selected range. The sender should be a menu item with a `representedObject` of type (`NSTextHighlightColorScheme`).

Relationships

Inherits From

`NSText`

Conforms To

`CVarArg`
`Copyable`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSAccessibilityElementProtocol`
`NSAccessibilityNavigableStaticText`
`NSAccessibilityProtocol`

`NSAccessibilityStaticText`
`NSAnimatablePropertyContainer`
`NSAppearanceCustomization`
`NSCandidateListTouchBarItemDelegate`
`NSChangeSpelling`
`NSCoding`
`NSColorChanging`
`NSDraggingDestination`
`NSDraggingSource`
`NSIgnoreMisspelledWords`
`NSMenuItemValidation`
`NSObjectProtocol`
`NSStandardKeyBindingResponding`
`NSTextContent`
`NSTextInput`
`NSTextInputClient`
`NSTextLayoutOrientationProvider`
`NSTouchBarDelegate`
`NSTouchBarProvider`
`NSUserActivityRestoring`
`NSUserInterfaceItemIdentification`
`NSUserInterfaceValidations`
`Sendable`
`SendableMetatype`

See Also

Text views

`class NSTextField`

Text the user can select or edit to send an action message to a target when the user presses the Return key.

`protocol NSTextFieldDelegate`

A protocol that a text field delegate can use to control its field editor action menu.

`protocol NSTextViewDelegate`

A set of optional methods that text view delegates can use to manage selection, set text attributes, work with the spell checker, and more.

`protocol NSTextDelegate`

A set of optional methods implemented by the delegate of an `NSText` object to edit text and change text formats.

`class NSText`

The most general programmatic interface for objects that manage text.