API Collection

TextKit

Manage text storage and perform custom layout of text-based content in your app's views.

Overview

TextKit provides several classes to control the layout of text, such as `NSTextContentStorage`, `NSTextLayoutManager`, and `NSTextContainer`.

Additionally, TextKit uses `NSAttributedString` objects extensively. The `NSTextStorage` class is a subclass of `NSMutableAttributedString`, and many of the TextKit classes, for example, the classes listed in `Formatted content`, focus on creating complex `NSAttributedString` instances. Use these classes to specify your text's format.

Most of the time, you can use TextKit to fine tune the formatting and layout of a `NSTextView` by modifying various properties of your view's layout manager, text container, or text storage objects in your app. If you need more control, you can also use TextKit to build your text controls.

Topics

Text management

`class NSTextContentStorage`

A concrete object for managing your view's text content and generating the text elements necessary for layout.

`class NSTextContentManager`

An abstract class that defines the interface and a default implementation for managing the text document contents.

`class NSAttributedString`

A string of text that manages data, layout, and stylistic information for ranges of characters to support rendering.

`class NSMutableAttributedString`

A mutable string with associated attributes (such as visual style, hyperlinks, or accessibility data) for portions of its text.

Formatting and attributes

`class NSParagraphStyle`

The paragraph or ruler attributes for an attributed string.

`class NSMutableParagraphStyle`

An object for changing the values of the subattributes in a paragraph style attribute.

`class NSTextTab`

A tab in a paragraph.

`class NSTextList`

A section of text that forms a single list.

`class NSTextTable`

An object that represents a text table as a whole.

`class NSTextTableBlock`

A text block that appears as a cell in a text table.

`class NSTextBlock`

A block of text laid out in a subregion of the text container.

Content elements

{} Enriching your text in text views

Add exclusion paths, text attachments, and text lists to your text, and render it with text views.

`class NSTextParagraph`

A class that represents a single paragraph backed by an attributed string as the contents.

`class NSTextListElement`

A class that represents a text list node.

`class NSTextElement`

An abstract base class that represents the smallest units of text layout such as paragraphs or attachments.

`protocol NSTextElementProvider`

A protocol the text content manager and its concrete subclasses conform to, which defines the interface for interacting with custom content types of a text document.

Location and selection

`class NSTextRange`

A class that represents a contiguous range between two locations inside document contents.

`class NSTextSelection`

A class that represents a single logical selection context that corresponds to an insertion point.

`class NSTextSelectionNavigation`

An interface you use to expose methods for obtaining results from actions performed on text selections.

`protocol NSTextLocation`

An interface you implement that represents an abstract location inside your document's content.

Layout

{} Using TextKit 2 to interact with text

Interact with text by managing text selection and inserting custom text elements.

`class NSTextLayoutManager`

The primary class that you use to manage text layout and presentation for custom text displays.

`class NSTextContainer`

A region where text layout occurs.

`class NSTextLayoutFragment`

A class that represents the layout fragment typically corresponding to a rendering surface, such as a layer or view subclass.

`class NSTextLineFragment`

A class that represents a line fragment as a single textual layout and rendering unit inside a text layout fragment.

`class NSTextViewportLayoutController`

Manages the layout process inside the viewport interacting with its delegate.

`protocol NSTextLayoutOrientationProvider`

A set of methods that define the orientation of text for an object.

Attachments

`class NSTextAttachment`

The values for the attachment characteristics of attributed strings and related objects.

`class NSTextAttachmentViewProvider`

A container object that associates a text attachment at a particular document location with a view object.

`class NSAdaptiveImageGlyph`

A data object for an emoji-like image that can appear in attributed text.

`protocol NSTextAttachmentContainer`

A set of methods that defines the interface to text attachment objects from a layout manager.

`protocol NSTextAttachmentLayout`

A set of methods that defines the interface to attachment objects from a text layout manager.

`class NSTextAttachmentCell`

An object that implements the functionality of the text attachment cell protocol.

`protocol NSTextAttachmentCellProtocol`

A set of methods that declares the interface for objects that draw text attachment icons and handle mouse events on their icons.

Glyphs

`typealias NSGlyph`

The type used to specify glyphs.

`protocol NSGlyphStorage`

A set of methods that a glyph storage object must implement to interact properly with `NSGlyphGenerator`.

`class NSGlyphGenerator`

An object that performs the initial, nominal glyph generation phase in the layout process.

`class NSGlyphInfo`

A glyph attribute in an attributed string.

Reserved Glyph Codes

These constants define reserved glyph codes.

`enum NSFontRenderingMode`

The font rendering mode.

TextKit 1

`class NSTextStorage`

The fundamental storage mechanism of TextKit that contains the text managed by the system.

`class NSLayoutManager`

An object that coordinates the layout and display of text characters.

`class NSATSTypesetter`

A concrete typesetter object that places glyphs during the text layout process.

`class NSTypesetter`

An abstract class that performs various type layout tasks.

See Also

Text

Text Display

Display text and check spelling.

Fonts

Manage the fonts used to display text.

Writing Tools

Add support for Writing Tools to your app’s text views.