API Collection

View Layout

Position and size views using a stack view or Auto Layout constraints.

Overview

When you design your app's interface, you position views and other interface elements in your app's windows and size them appropriately. However, the size and position of those views may need to change at runtime for a few reasons:

- The user resizes the window containing your views.
- The user's language choice requires size changes for labels and text-based views.

`NSStackView` objects adjust the position of their contained views automatically when interface dimensions change. Alternatively, Auto Layout constraints let you specify the rules that determine the precise size and position of the views in your interface

---

Topics

Stack View

`class NSStackView`

A view that arranges an array of views horizontally or vertically and updates their placement and sizing when the window size changes.

Auto Layout Constraints

`class NSLayoutConstraint`

The relationship between two user interface objects that must be satisfied by the constraint-based layout system.

Layout Guides

`class` `NSLayoutGuide`

A rectangular area that can interact with Auto Layout.

`class` `NSLayoutDimension`

A factory class for creating size-based layout constraint objects using a fluent API.

Anchors

`class` `NSLayoutAnchor`

A factory class for creating layout constraint objects using a fluent API.

`class` `NSLayoutXAxisAnchor`

A factory class for creating horizontal layout constraint objects using a fluent API.

`class` `NSLayoutYAxisAnchor`

A factory class for creating vertical layout constraint objects using a fluent API.

View Compression

`protocol` `NSUserInterfaceCompression`

A protocol that describes how a UI control should redisplay when space is restricted.

See Also

User Interface

Views and Controls

Present your content onscreen and handle user input and events.

View Management

Manage your user interface, including the size and position of views in a window.

- Appearance Customization

  Add Dark Mode support to your app, and use appearance proxies to modify your UI.

- Animation

  Animate your views and other content to create a more engaging experience for users.

- Windows, Panels, and Screens

  Organize your view hierarchies and facilitate their display onscreen.

- Sound, Speech, and Haptics

  Play sounds and haptic feedback, and incorporate speech recognition and synthesis into your interface.

- Supporting Continuity Camera in Your Mac App

  Incorporate scanned documents and pictures from a user's iPhone, iPad, or iPod touch into your Mac app using Continuity Camera.