SwiftUI views for widgets

Present your app's content in widgets with SwiftUI views.

Overview

Widgets can use many, but not all, SwiftUI views to present content. Use the views listed below to implement your widget's view.

> **Note**
>
> Widgets can't use UIKit or AppKit views wrapped with `UIViewRepresentable` or `NSViewRepresentable`.

Topics

Displaying text

Displaying dynamic dates in widgets

Show up-to-date, time-based information in your widget even when it isn't running.

`@frozen struct Text`

A view that displays one or more lines of read-only text.

Showing images

`@frozen struct Image`

A view that displays an image.

Adding interaction

Adding interactivity to widgets and Live Activities

Include buttons or toggles in a widget or Live Activity to offer app functionality without launching the app.

`struct Button<Label> where Label : View`

A control that initiates an action.

`struct Toggle<Label> where Label : View`

A control that toggles between on and off states.

Adding labels and links

`struct Label<Title, Icon> where Title : View, Icon : View`

A standard label for user interface items, consisting of an icon with a title.

`@MainActor @preconcurrency struct Link<Label> where Label : View`

A control for navigating to a URL.

Stacking views

`@frozen struct HStack<Content> where Content : View`

A view that arranges its subviews in a horizontal line.

`@frozen struct VStack<Content> where Content : View`

A view that arranges its subviews in a vertical line.

`@frozen struct ZStack<Content> where Content : View`

A view that overlays its subviews, aligning them in both axes.

`struct LazyHStack<Content> where Content : View`

A view that arranges its children in a line that grows horizontally, creating items only as needed.

`struct LazyVStack<Content> where Content : View`

A view that arranges its children in a line that grows vertically, creating items only as needed.

Arranging views in grids

`struct LazyHGrid<Content> where Content : View`

A container view that arranges its child views in a grid that grows horizontally, creating items only as needed.

`struct LazyVGrid<Content> where Content : View`

A container view that arranges its child views in a grid that grows vertically, creating items only as needed.

`struct GridItem`

A description of a row or a column in a lazy grid.

Enumerating lists

`struct ForEach<Data, ID, Content> where Data : RandomAccessCollection, ID : Hashable`

A structure that computes views on demand from an underlying collection of identified data.

Grouping views

`@frozen struct Group<Content>`

A type that collects multiple instances of a content type — like views, scenes, or commands — into a single unit.

`struct GroupBox<Label, Content> where Label : View, Content : View`

A stylized view, with an optional label, that visually collects a logical grouping of content.

`struct Section<Parent, Content, Footer>`

A container view that you can use to add hierarchy within certain views.

Representing hierarchies

`struct OutlineGroup<Data, ID, Parent, Leaf, Subgroup> where Data : RandomAccessCollection, ID : Hashable`

A structure that computes views and disclosure groups on demand from an underlying collection of tree-structured, identified data.

Adding spacers and dividers

`@frozen struct Spacer`

A flexible space that expands along the major axis of its containing stack layout, or on both axes if not contained in a stack.

`struct Divider`

A visual element that can be used to separate other content.

Handling conditional views

`@frozen struct EmptyView`

A view that doesn't contain any content.

`@frozen struct EquatableView<Content> where Content : Equatable, Content : View`

A view type that compares itself against its previous value and prevents its child updating if its new value is the same as its old value.

Displaying shapes

`@frozen struct Rectangle`

A rectangular shape aligned inside the frame of the view containing it.

`@frozen struct RoundedRectangle`

A rectangular shape with rounded corners, aligned inside the frame of the view containing it.

`@frozen struct Circle`

A circle centered on the frame of the view containing it.

`@frozen struct Ellipse`

An ellipse aligned inside the frame of the view containing it.

`@frozen struct Capsule`

A capsule shape aligned inside the frame of the view containing it.

`@frozen struct Path`

The outline of a 2D shape.

Transforming views

`@frozen struct ScaledShape<Content> where Content : Shape`

A shape with a scale transform applied to it.

`@frozen struct RotatedShape<Content> where Content : Shape`

A shape with a rotation transform applied to it.

`@frozen struct OffsetShape<Content> where Content : Shape`

A shape with a translation offset transform applied to it.

`@frozen struct TransformedShape<Content> where Content : Shape`

A shape with an affine transform applied to it.

`@frozen struct ContainerRelativeShape`

A shape that is replaced by an inset version of the current container shape. If no container shape was defined, is replaced by a rectangle.

Styling views

`@frozen struct Color`

A representation of a color that adapts to a given context.

`@frozen struct ImagePaint`

A shape style that fills a shape by repeating a region of an image.

`@frozen struct Gradient`

A color gradient represented as an array of color stops, each having a parametric location value.

`@frozen struct LinearGradient`

A linear gradient.

`@frozen struct AngularGradient`

An angular gradient.

`@frozen struct RadialGradient`

A radial gradient.

`@frozen struct ForegroundStyle`

The foreground style in the current context.

`@frozen struct FillStyle`

A style for rasterizing vector shapes.

`@frozen struct BackgroundStyle`

The background style in the current context.

`struct` SelectionShapeStyle

A style used to visually indicate selection following platform conventional colors and behaviors.

`struct` SeparatorShapeStyle

A style appropriate for foreground separator or border lines.

`@frozen struct` StrokeStyle

The characteristics of a stroke that traces a path.

Creating 2D graphics

`struct` Canvas<Symbols> `where` Symbols : View

A view type that supports immediate mode drawing.

Managing view geometry

`struct` GeometryProxy

A proxy for access to the size and coordinate space (for anchor resolution) of the container view.

`@frozen struct` GeometryReader<Content> `where` Content : View

A container view that defines its content as a function of its own size and coordinate space.

`@frozen struct` ProjectionTransform

Substituting views

`@frozen struct` AnyView

A type-erased view.

`@frozen struct` TupleView<T>

A View created from a swift tuple of View values.

See Also

Presentation

Creating views for widgets, Live Activities, and watch complications

Implement glanceable views with WidgetKit and SwiftUI.