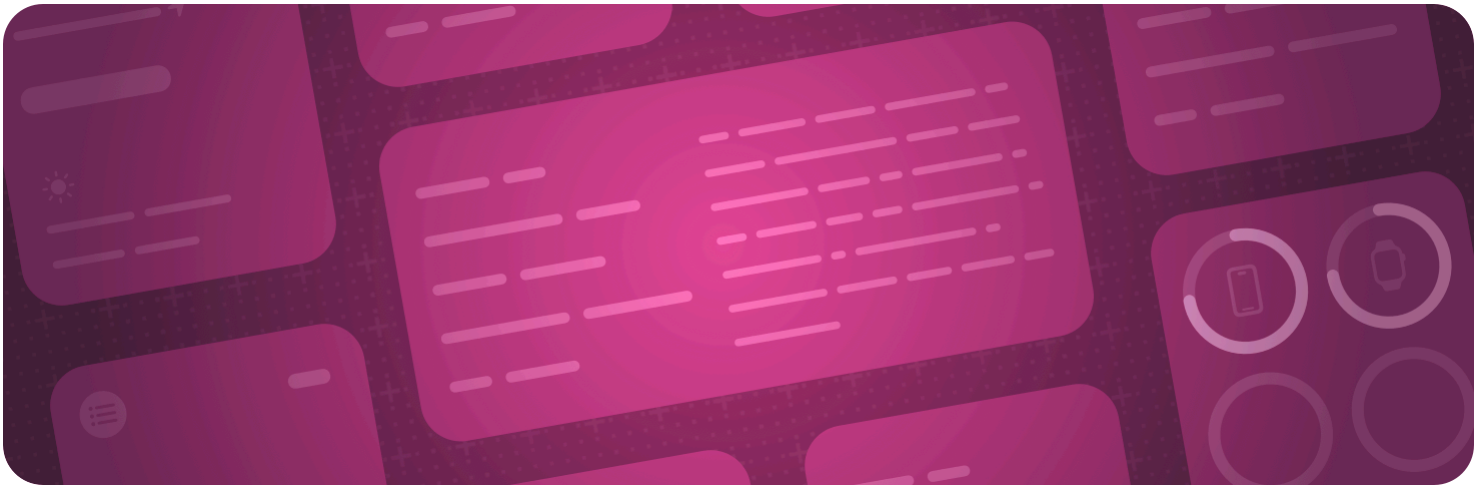
API Collection

App extensions

Extend your app's basic functionality to other parts of the system, like by adding a Widget.

Overview

Use SwiftUI along with WidgetKit to add widgets to your app.



Widgets provide quick access to relevant content from your app. Define a structure that conforms to the `Widget` protocol, and declare a view hierarchy for the widget. Configure the views inside the widget as you do other SwiftUI views, using view modifiers, including a few widget-specific modifiers.

For design guidance, see Widgets in the Human Interface Guidelines.

Topics

Creating widgets

{} Building Widgets Using WidgetKit and SwiftUI

Create widgets to show your app's content on the Home screen, with custom intents for user-customizable settings.

Creating a widget extension

Display your app's content in a convenient, informative widget on various devices.

Keeping a widget up to date

Plan your widget's timeline to show timely, relevant information using dynamic views, and update the timeline when things change.

Making a configurable widget

Give people the option to customize their widgets by adding a custom app intent to your project.

`protocol Widget`

The configuration and content of a widget to display on the Home screen or in Notification Center.

`protocol WidgetBundle`

A container used to expose multiple widgets from a single widget extension.

`struct LimitedAvailabilityConfiguration`

A type-erased widget configuration.

`protocol WidgetConfiguration`

A type that describes a widget's content.

`struct EmptyWidgetConfiguration`

An empty widget configuration.

Composing control widgets

`protocol ControlWidget`

The configuration and content of a control widget to display in system spaces such as Control Center, the Lock Screen, and the Action Button.

`protocol ControlWidgetConfiguration`

A type that describes a control widget's content.

`struct EmptyControlWidgetConfiguration`

An empty control widget configuration.

`struct ControlWidgetConfigurationBuilder`

A custom attribute that constructs a control widget's body.

`protocol ControlWidgetTemplate`

A type that describes a control widget's content.

`struct EmptyControlWidgetTemplate`

An empty control widget template.

`struct ControlWidgetTemplateBuilder`

A custom attribute that constructs a control widget template's body.

`func controlWidgetActionHint(_:)`

The action hint of the control described by the modified label.

`func controlWidgetStatus(_:)`

The status of the control described by the modified label.

Labeling a widget

`func widgetLabel(_:)`

Returns a localized text label that displays additional content outside the accessory family widget's main SwiftUI view.

`func widgetLabel<Label>(label: () -> Label) -> some View`

Creates a label for displaying additional content outside an accessory family widget's main SwiftUI view.

Styling a widget group

`func accessoryWidgetGroupStyle(AccessoryWidgetGroupStyle) -> some View`

The view modifier that can be applied to `AccessoryWidgetGroup` to specify the shape the three content views will be masked with. The value of `style` is set to `.automatic`, which is `.circular` by default.

Controlling the accented group

`func widgetAccentable(Bool) -> some View`

Adds the view and all of its subviews to the accented group.

Managing placement in the Dynamic Island

`func dynamicIsland(verticalPlacement: DynamicIslandExpandedRegion VerticalPlacement) -> some View`

Specifies the vertical placement for a view of an expanded Live Activity that appears in the Dynamic Island.

See Also

App structure

- App organization

  Define the entry point and top-level structure of your app.

- Scenes

  Declare the user interface groupings that make up the parts of your app.

- Windows

  Display user interface content in a window or a collection of windows.

- Immersive spaces

  Display unbounded content in a person's surroundings.

- Documents

  Enable people to open and manage documents.

- Navigation

  Enable people to move between different parts of your app's view hierarchy within a scene.

- Modal presentations

  Present content in a separate view that offers focused interaction.

- Toolbars

  Provide immediate access to frequently used commands and controls.

Search

Enable people to search for text or other content within your app.