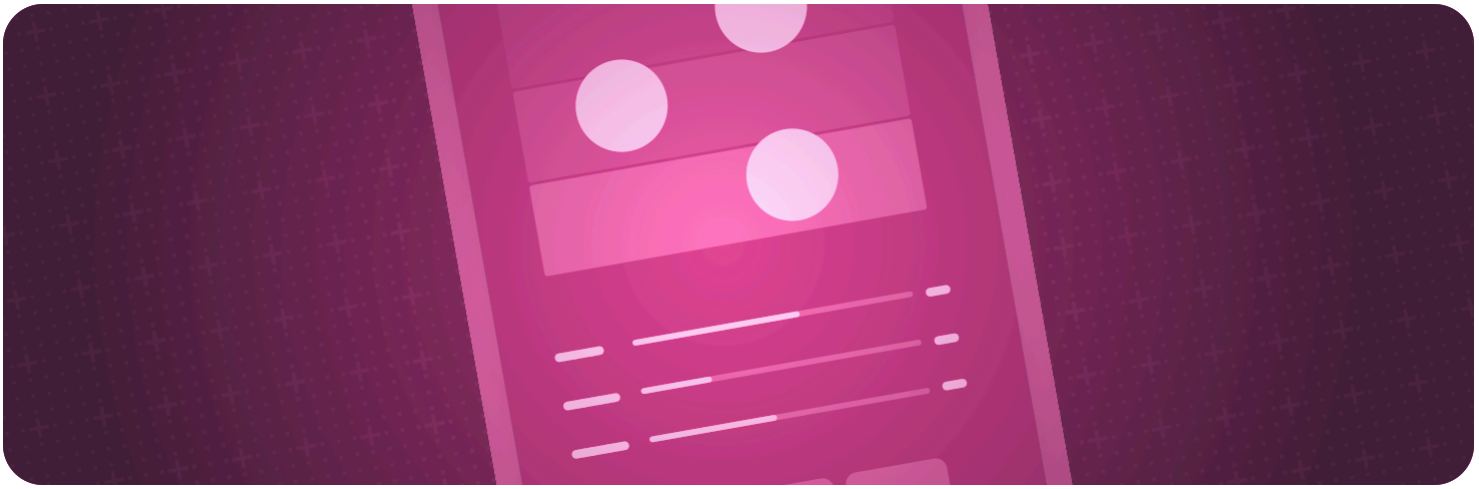
API Collection

Custom layout

Place views in custom arrangements and create animated transitions between layout types.

Overview

You can create complex view layouts using the built-in layout containers and layout view modifiers that SwiftUI provides. However, if you need behavior that you can't achieve with the built-in layout tools, create a custom layout container type using the `Layout` protocol. A container that you define asks for the sizes of all its subviews, and then indicates where to place the subviews within its own bounds.



You can also create animated transitions among layout types that conform to the `Layout` procotol, including both built-in and custom layouts.

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

Topics

Creating a custom layout container

{} Composing custom layouts with SwiftUI

Arrange views in your app's interface using layout tools that SwiftUI provides.

`protocol Layout`

A type that defines the geometry of a collection of views.

`struct LayoutSubview`

A proxy that represents one subview of a layout.

`struct LayoutSubviews`

A collection of proxy values that represent the subviews of a layout view.

Configuring a custom layout

`struct LayoutProperties`

Layout-specific properties of a layout container.

`struct ProposedViewSize`

A proposal for the size of a view.

`struct ViewSpacing`

A collection of the geometric spacing preferences of a view.

Associating values with views in a custom layout

`func layoutValue<K>(key: K.Type, value: K.Value) -> some View`

Associates a value with a custom layout property.

`protocol LayoutValueKey`

A key for accessing a layout value of a layout container's subviews.

Transitioning between layout types

`struct AnyLayout`

A type-erased instance of the layout protocol.

`struct HStackLayout`

A horizontal container that you can use in conditional layouts.

`struct VStackLayout`

A vertical container that you can use in conditional layouts.

`struct ZStackLayout`

An overlaying container that you can use in conditional layouts.

`struct GridLayout`

A grid that you can use in conditional layouts.

See Also

View layout

- Layout fundamentals

  Arrange views inside built-in layout containers like stacks and grids.

- Layout adjustments

  Make fine adjustments to alignment, spacing, padding, and other layout parameters.

- Lists

  Display a structured, scrollable column of information.

- Tables

  Display selectable, sortable data arranged in rows and columns.

- View groupings

  Present views in different kinds of purpose-driven containers, like forms or control groups.

- Scroll views

  Enable people to scroll to content that doesn’t fit in the current display.