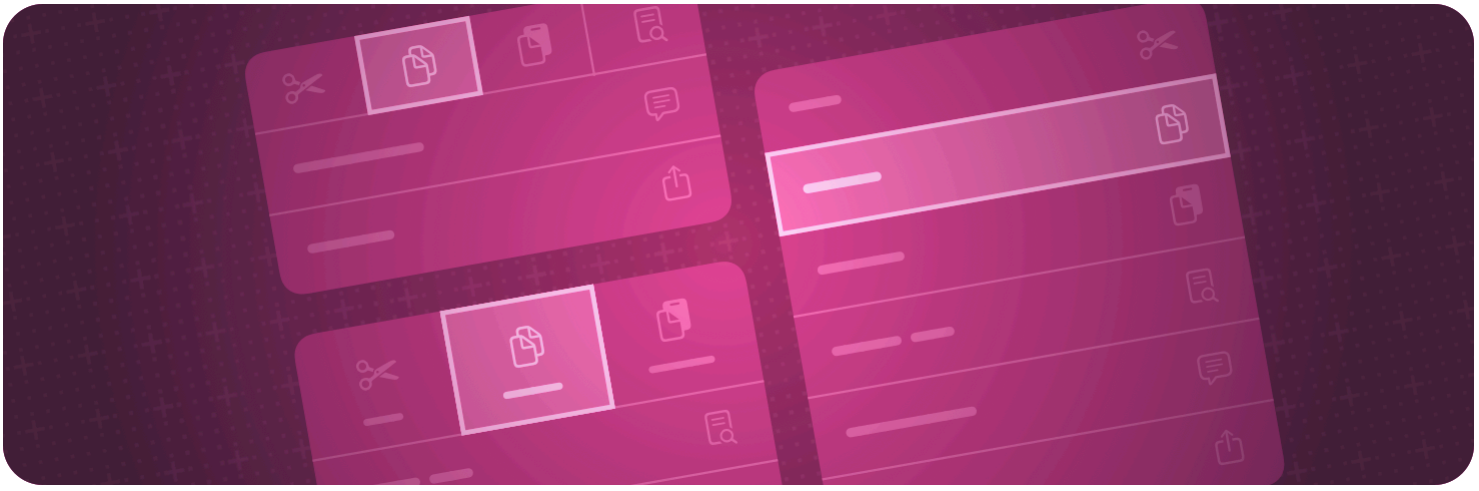
API Collection

View styles

Apply built-in and custom appearances and behaviors to different types of views.

Overview

SwiftUI defines built-in styles for certain kinds of views and automatically selects the appropriate style for a particular presentation context. For example, a `Label` might appear as an icon, a string title, or both, depending on factors like the platform, whether the view appears in a toolbar, and so on.



You can override the automatic style by using one of the style view modifiers. These modifiers typically propagate throughout a container view, so that you can wrap a view hierarchy in a style modifier to affect all the views of the given type within the hierarchy.

Any of the style protocols that define a `makeBody(configuration:)` method, like `ToggleStyle`, also enable you to define custom styles. Create a type that conforms to the corresponding style protocol and implement its `makeBody(configuration:)` method. Then apply the new style using a style view modifier exactly like a built-in style.

Topics

Styling views with Liquid Glass

Applying Liquid Glass to custom views

Configure, combine, and morph views using Liquid Glass effects.

{} Landmarks: Building an app with Liquid Glass

Enhance your app experience with system-provided and custom Liquid Glass.

`func glassEffect(Glass, in: some Shape) -> some View`

Applies the Liquid Glass effect to a view.

`func interactive(Bool) -> Glass`

Returns a copy of the structure configured to be interactive.

`struct GlassEffectContainer`

A view that combines multiple Liquid Glass shapes into a single shape that can morph individual shapes into one another.

`struct GlassEffectTransition`

A structure that describes changes to apply when a glass effect is added or removed from the view hierarchy.

`struct GlassButtonStyle`

A button style that applies glass border artwork based on the button's context.

`struct GlassProminentButtonStyle`

A button style that applies prominent glass border artwork based on the button's context.

`struct DefaultGlassEffectShape`

The default shape applied by glass effects, a capsule.

Styling buttons

`func buttonStyle(_:)`

Sets the style for buttons within this view to a button style with a custom appearance and standard interaction behavior.

`protocol` `ButtonStyle`

A type that applies standard interaction behavior and a custom appearance to all buttons within a view hierarchy.

`struct` `ButtonStyleConfiguration`

The properties of a button.

`protocol` `PrimitiveButtonStyle`

A type that applies custom interaction behavior and a custom appearance to all buttons within a view hierarchy.

`struct` `PrimitiveButtonStyleConfiguration`

The properties of a button.

`func signInWithAppleButtonStyle(SignInWithAppleButton.Style) -> some View`

Sets the style used for displaying the control (see `SignInWithAppleButton.Style`).

Styling pickers

`func pickerStyle<S>(S) -> some View`

Sets the style for pickers within this view.

`protocol` `PickerStyle`

A type that specifies the appearance and interaction of all pickers within a view hierarchy.

`func datePickerStyle<S>(S) -> some View`

Sets the style for date pickers within this view.

`protocol` `DatePickerStyle`

A type that specifies the appearance and interaction of all date pickers within a view hierarchy.

Styling menus

`func menuStyle<S>(S) -> some View`

Sets the style for menus within this view.

`protocol` `MenuStyle`

A type that applies standard interaction behavior and a custom appearance to all menus within a view hierarchy.

`struct MenuStyleConfiguration`

A configuration of a menu.

Styling toggles

`func toggleStyle<S>(S) -> some View`

Sets the style for toggles in a view hierarchy.

`protocol ToggleStyle`

The appearance and behavior of a toggle.

`struct ToggleStyleConfiguration`

The properties of a toggle instance.

Styling indicators

`func gaugeStyle<S>(S) -> some View`

Sets the style for gauges within this view.

`protocol GaugeStyle`

Defines the implementation of all gauge instances within a view hierarchy.

`struct GaugeStyleConfiguration`

The properties of a gauge instance.

`func progressViewStyle<S>(S) -> some View`

Sets the style for progress views in this view.

`protocol ProgressViewStyle`

A type that applies standard interaction behavior to all progress views within a view hierarchy.

`struct ProgressViewStyleConfiguration`

The properties of a progress view instance.

Styling views that display text

`func labelStyle<S>(S) -> some View`

Sets the style for labels within this view.

`protocol LabelStyle`

A type that applies a custom appearance to all labels within a view.

`struct LabelStyleConfiguration`

The properties of a label.

`func textFieldStyle<S>(S) -> some View`

Sets the style for text fields within this view.

`protocol TextFieldStyle`

A specification for the appearance and interaction of a text field.

`func textEditorStyle(some TextEditorStyle) -> some View`

Sets the style for text editors within this view.

`protocol TextEditorStyle`

A specification for the appearance and interaction of a text editor.

`struct TextEditorStyleConfiguration`

The properties of a text editor.

Styling collection views

`func listStyle<S>(S) -> some View`

Sets the style for lists within this view.

`protocol ListStyle`

A protocol that describes the behavior and appearance of a list.

`func tableStyle<S>(S) -> some View`

Sets the style for tables within this view.

`protocol TableStyle`

A type that applies a custom appearance to all tables within a view.

`struct TableStyleConfiguration`

The properties of a table.

`func disclosureGroupStyle<S>(S) -> some View`

Sets the style for disclosure groups within this view.

`protocol DisclosureGroupStyle`

A type that specifies the appearance and interaction of disclosure groups within a view hierarchy.

Styling navigation views

`func navigationSplitViewStyle<S>(S) -> some View`

Sets the style for navigation split views within this view.

`protocol NavigationSplitViewStyle`

A type that specifies the appearance and interaction of navigation split views within a view hierarchy.

`func tabViewStyle<S>(S) -> some View`

Sets the style for the tab view within the current environment.

`protocol TabViewStyle`

A specification for the appearance and interaction of a tab view.

Styling groups

`func controlGroupStyle<S>(S) -> some View`

Sets the style for control groups within this view.

`protocol ControlGroupStyle`

Defines the implementation of all control groups within a view hierarchy.

`struct ControlGroupStyleConfiguration`

The properties of a control group.

`func formStyle<S>(S) -> some View`

Sets the style for forms in a view hierarchy.

`protocol FormStyle`

The appearance and behavior of a form.

`struct FormStyleConfiguration`

The properties of a form instance.

`func groupBoxStyle<S>(S) -> some View`

Sets the style for group boxes within this view.

`protocol GroupBoxStyle`

A type that specifies the appearance and interaction of all group boxes within a view hierarchy.

`struct GroupBoxStyleConfiguration`

The properties of a group box instance.

`func indexViewStyle<S>(S) -> some View`

Sets the style for the index view within the current environment.

`protocol IndexViewStyle`

Defines the implementation of all `IndexView` instances within a view hierarchy.

`func labeledContentStyle<S>(S) -> some View`

Sets a style for labeled content.

`protocol LabeledContentStyle`

The appearance and behavior of a labeled content instance..

`struct LabeledContentStyleConfiguration`

The properties of a labeled content instance.

Styling windows from a view inside the window

`func presentedWindowStyle<S>(S) -> some View`

Sets the style for windows created by interacting with this view.

`func presentedWindowToolbarStyle<S>(S) -> some View`

Sets the style for the toolbar in windows created by interacting with this view.

Adding a glass background on views in visionOS

`func glassBackgroundEffect(displayMode: GlassBackgroundDisplayMode) -> some View`

Fills the view's background with an automatic glass background effect and container-relative rounded rectangle shape.

`func glassBackgroundEffect<S>(in: S, displayMode: GlassBackgroundDisplayMode) -> some View`

Fills the view's background with an automatic glass background effect and a shape that you specify.

`enum` `GlassBackgroundDisplayMode`

The display mode of a glass background.

`protocol` `GlassBackgroundEffect`

A specification for the appearance of a glass background.

`struct` `AutomaticGlassBackgroundEffect`

The automatic glass background effect.

`struct` `GlassBackgroundEffectConfiguration`

A configuration used to build a custom effect.

`struct` `FeatheredGlassBackgroundEffect`

The feathered glass background effect.

`struct` `PlateGlassBackgroundEffect`

The plate glass background effect.

See Also

Views

- View fundamentals

  Define the visual elements of your app using a hierarchy of views.

- View configuration

  Adjust the characteristics of views in a hierarchy.

- Animations

  Create smooth visual updates in response to state changes.

- Text input and output

  Display formatted text and get text input from the user.

- Images

  Add images and symbols to your app's user interface.

- Controls and indicators

  Display values and get user selections.

- Menus and commands

  Provide space-efficient, context-dependent access to commands and controls.

- Shapes

  Trace and fill built-in and custom shapes with a color, gradient, or other pattern.

- Drawing and graphics

  Enhance your views with graphical effects and customized drawings.