API Collection

Previews in Xcode

Generate dynamic, interactive previews of your custom views.

Overview

When you create a custom `View` with SwiftUI, Xcode can display a preview of the view's content that stays up-to-date as you make changes to the view's code. You use one of the preview macros — like `Preview(_:body:)` — to tell Xcode what to display. Xcode shows the preview in a canvas beside your code.

Different preview macros enable different kinds of configuration. For example, you can add traits that affect the preview's appearance using the `Preview(_:traits:_:body:)` macro or add custom viewpoints for the preview using the `Preview(_:traits:body:cameras:)` macro. You can also check how your view behaves inside a specific scene type. For example, in visionOS you can use the `Preview(_:immersionStyle:traits:body:)` macro to preview your view inside an `ImmersiveSpace`.

You typically rely on preview macros to create previews in your code. However, if you can't get the behavior you need using a preview macro, you can use the `PreviewProvider` protocol and its associated supporting types to define and configure a preview.

Topics

Essentials

Previewing your app’s interface in Xcode

Iterate designs quickly and preview your apps’ displays across different Apple devices.

Creating a preview

`macro Preview(String?, body: () -> any View)`

Creates a preview of a SwiftUI view.

`macro Preview(String?, traits: PreviewTrait<Preview.ViewTraits>, PreviewTrait<Preview.ViewTraits>..., body: () -> any View)`

Creates a preview of a SwiftUI view using the specified traits.

`macro Preview(String?, traits: PreviewTrait<Preview.ViewTraits>..., body: () -> any View, cameras: () -> [PreviewCamera])`

Creates a preview of a SwiftUI view using the specified traits and custom viewpoints.

Creating a preview in the context of a scene

`macro Preview<Style>(String?, immersionStyle: Style, traits: PreviewTrait<Preview.ViewTraits>..., body: () -> any View)`

Creates a preview of a SwiftUI view in an immersive space.

`macro Preview<Style>(String?, immersionStyle: Style, traits: PreviewTrait<Preview.ViewTraits>..., body: () -> any View, cameras: () -> [PreviewCamera])`

Creates a preview of a SwiftUI view in an immersive space with custom viewpoints.

`macro Preview<Style>(String?, windowStyle: Style, traits: PreviewTrait<Preview.ViewTraits>..., body: () -> any View)`

Creates a preview of a SwiftUI view in a window.

`macro Preview<Style>(String?, windowStyle: Style, traits: PreviewTrait<Preview.ViewTraits>..., body: () -> any View, cameras: () -> [PreviewCamera])`

Creates a preview of a SwiftUI view in a window with custom viewpoints.

Defining a preview

`macro Previewable()`

Tag allowing a dynamic property to appear inline in a preview.

`protocol PreviewProvider`

A type that produces view previews in Xcode.

`enum PreviewPlatform`

Platforms that can run the preview.

`func previewDisplayName(String?) -> some View`

Sets a user visible name to show in the canvas for a preview.

`protocol PreviewModifier`

A type that defines an environment in which previews can appear.

`struct PreviewModifierContent`

The type-erased content of a preview.

Customizing a preview

`func previewDevice(PreviewDevice?) -> some View`

Overrides the device for a preview.

`struct PreviewDevice`

A simulator device that runs a preview.

`func previewLayout(PreviewLayout) -> some View`

Overrides the size of the container for the preview.

`func previewInterfaceOrientation(InterfaceOrientation) -> some View`

Overrides the orientation of the preview.

`struct InterfaceOrientation`

The orientation of the interface from the user’s perspective.

Setting a context

`func previewContext<C>(C) -> some View`

Declares a context for the preview.

`protocol PreviewContext`

A context type for use with a preview.

`protocol PreviewContextKey`

A key type for a preview context.

Building in debug mode

`struct DebugReplaceableView`

Erases view opaque result types in debug builds.

See Also

Tool support

Xcode library customization

Expose custom views and modifiers in the Xcode library.

Performance analysis

Measure and improve your app’s responsiveness.