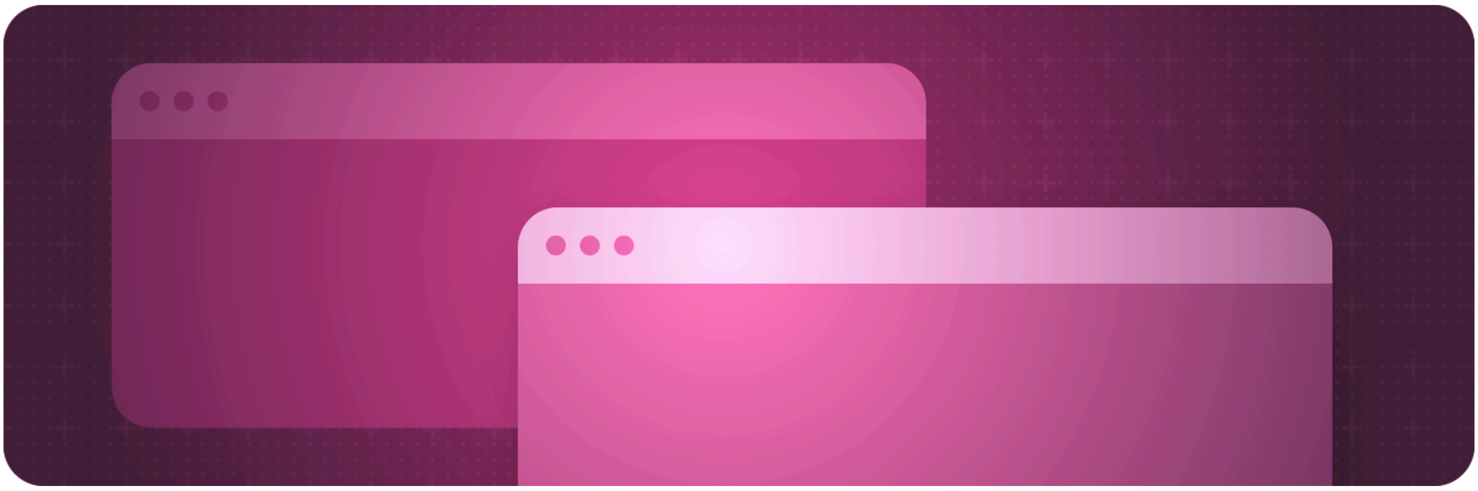
API Collection

Windows

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

Overview

The most common way to present a view hierarchy in your app's interface is with a `WindowGroup`, which produces a platform-specific behavior and appearance.



On platforms that support it, people can open multiple windows from the group simultaneously. Each window relies on the same root view definition, but retains its own view state. On some platforms, you can also supplement your app's user interface with a single-instance window using the `Window` scene type.

Configure windows using scene modifiers that you add to the window declaration, like `windowStyle(_:)` or `defaultPosition(_:)`. You can also indicate how to configure new windows that you present from a view hierarchy by adding the `presentedWindowStyle(_:)` view modifier to a view in the hierarchy.

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

Topics

Essentials

{} Customizing window styles and state-restoration behavior in macOS

Configure how your app's windows look and function in macOS to provide an engaging and more coherent experience.

{} Bringing multiple windows to your SwiftUI app

Compose rich views by reacting to state changes and customize your app's scene presentation and behavior on iPadOS and macOS.

Creating windows

`struct WindowGroup`

A scene that presents a group of identically structured windows.

`struct Window`

A scene that presents its content in a single, unique window.

`struct UtilityWindow`

A specialized window scene that provides secondary utility to the content of the main scenes of an application.

`protocol WindowStyle`

A specification for the appearance and interaction of a window.

`func windowStyle<S>(S) -> some Scene`

Sets the style for windows created by this scene.

Styling the associated toolbar

`func windowToolbarStyle<S>(S) -> some Scene`

Sets the style for the toolbar defined within this scene.

`func windowToolbarLabelStyle(Binding<ToolbarLabelStyle>) -> some Scene`

Sets the label style of items in a toolbar and enables user customization.

`func windowToolbarLabelStyle(fixed: ToolbarLabelStyle) -> some Scene`

Sets the label style of items in a toolbar.

`protocol WindowToolbarStyle`

A specification for the appearance and behavior of a window's toolbar.

Opening windows

Presenting windows and spaces

Open and close the scenes that make up your app's interface.

`var supportsMultipleWindows: Bool`

A Boolean value that indicates whether the current platform supports opening multiple windows.

`var openWindow: OpenWindowAction`

A window presentation action stored in a view's environment.

`struct OpenWindowAction`

An action that presents a window.

`struct PushWindowAction`

An action that opens the requested window in place of the window the action is called from.

Closing windows

`var dismissWindow: DismissWindowAction`

A window dismissal action stored in a view's environment.

`struct DismissWindowAction`

An action that dismisses a window associated to a particular scene.

`var dismiss: DismissAction`

An action that dismisses the current presentation.

`struct DismissAction`

An action that dismisses a presentation.

`struct DismissBehavior`

Programmatic window dismissal behaviors.

Sizing a window

Positioning and sizing windows

Influence the initial geometry of windows that your app presents.

`func defaultSize(_:)`

Sets a default size for a window.

`func defaultSize(width: CGFloat, height: CGFloat) -> some Scene`

Sets a default width and height for a window.

`func defaultSize(width: CGFloat, height: CGFloat, depth: CGFloat) -> some Scene`

Sets a default size for a volumetric window.

`func defaultSize(Size3D, in: UnitLength) -> some Scene`

Sets a default size for a volumetric window.

`func defaultSize(width: CGFloat, height: CGFloat, depth: CGFloat, in: UnitLength) -> some Scene`

Sets a default size for a volumetric window.

`func windowResizability(WindowResizability) -> some Scene`

Sets the kind of resizability to use for a window.

`struct WindowResizability`

The resizability of a window.

`func windowIdealSize(WindowIdealSize) -> some Scene`

Specifies how windows derived form this scene should determine their size when zooming.

`struct WindowIdealSize`

A type which defines the size a window should use when zooming.

Positioning a window

`func defaultPosition(UnitPoint) -> some Scene`

Sets a default position for a window.

`struct WindowLevel`

The level of a window.

`func windowLevel(WindowLevel) -> some Scene`

Sets the window level of this scene.

`struct WindowLayoutRoot`

A proxy which represents the root contents of a window.

`struct WindowPlacement`

A type which represents a preferred size and position for a window.

`func defaultWindowPlacement((WindowLayoutRoot, WindowPlacementContext) -> WindowPlacement) -> some Scene`

Defines a function used for determining the default placement of windows.

`func windowIdealPlacement((WindowLayoutRoot, WindowPlacementContext) -> WindowPlacement) -> some Scene`

Provides a function which determines a placement to use when windows of a scene zoom.

`struct WindowPlacementContext`

A type which represents contextual information used for sizing and positioning windows.

`struct WindowProxy`

The proxy for an open window in the app.

`struct DisplayProxy`

A type which provides information about display hardware.

Configuring window visibility

`struct WindowVisibilityToggle`

A specialized button for toggling the visibility of a window.

`func defaultLaunchBehavior(SceneLaunchBehavior) -> some Scene`

Sets the default launch behavior for this scene.

`func restorationBehavior(SceneRestorationBehavior) -> some Scene`

Sets the restoration behavior for this scene.

`struct SceneLaunchBehavior`

The launch behavior for a scene.

`struct SceneRestorationBehavior`

The restoration behavior for a scene.

`func persistentSystemOverlays(Visibility) -> some Scene`

Sets the preferred visibility of the non-transient system views overlaying the app.

`func windowToolbarFullScreenVisibility(WindowToolbarFullScreenVisibility) -> some View`

Configures the visibility of the window toolbar when the window enters full screen mode.

`struct WindowToolbarFullScreenVisibility`

The visibility of the window toolbar with respect to full screen mode.

Managing window behavior

`struct WindowManagerRole`

Options for defining how a scene’s windows behave when used within a managed window context, such as full screen mode and Stage Manager.

`func windowManagerRole(WindowManagerRole) -> some Scene`

Configures the role for windows derived from `self` when participating in a managed window context, such as full screen or Stage Manager.

`struct WindowInteractionBehavior`

Options for enabling and disabling window interaction behaviors.

`func windowDismissBehavior(WindowInteractionBehavior) -> some View`

Configures the dismiss functionality for the window enclosing `self`.

`func windowFullScreenBehavior(WindowInteractionBehavior) -> some View`

Configures the full screen functionality for the window enclosing `self`.

`func windowMinimizeBehavior(WindowInteractionBehavior) -> some View`

Configures the minimize functionality for the window enclosing `self`.

`func windowResizeBehavior(WindowInteractionBehavior) -> some View`

Configures the resize functionality for the window enclosing `self`.

`func windowBackgroundDragBehavior(WindowInteractionBehavior) -> some Scene`

Configures the behavior of dragging a window by its background.

Interacting with volumes

`func onVolumeViewpointChange(updateStrategy: VolumeViewpointUpdateStrategy, initial: Bool, (Viewpoint3D, Viewpoint3D) -> Void) -> some View`

Adds an action to perform when the viewpoint of the volume changes.

`func supportedVolumeViewpoints(SquareAzimuth.Set) -> some View`

Specifies which viewpoints are supported for the window bar and ornaments in a volume.

`struct VolumeViewpointUpdateStrategy`

A type describing when the action provided to `onVolumeViewpointChange(updateStrategy:initial:_:)` should be called.

`struct Viewpoint3D`

A type describing what direction something is being viewed from.

`enum SquareAzimuth`

A type describing what direction something is being viewed from along the horizontal plane and snapped to 4 directions.

`struct WorldAlignmentBehavior`

A type representing the world alignment behavior for a scene.

`func volumeWorldAlignment(WorldAlignmentBehavior) -> some Scene`

Specifies how a volume should be aligned when moved in the world.

`struct WorldScalingBehavior`

Specifies the scaling behavior a window should have within the world.

`func defaultWorldScaling(WorldScalingBehavior) -> some Scene`

Specify the world scaling behavior for the window.

`struct WorldScalingCompensation`

Indicates whether returned metrics will take dynamic scaling into account.

`var worldTrackingLimitations: Set<WorldTrackingLimitation>`

The current limitations of the device tracking the user's surroundings.

`struct WorldTrackingLimitation`

A structure to represent limitations of tracking the user's surroundings.

`struct SurfaceSnappingInfo`

A type representing information about the window scenes snap state.

Deprecated Types

~~`enum ControlActiveState`~~

The active appearance expected of controls in a window.

Deprecated

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.

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.

App extensions

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