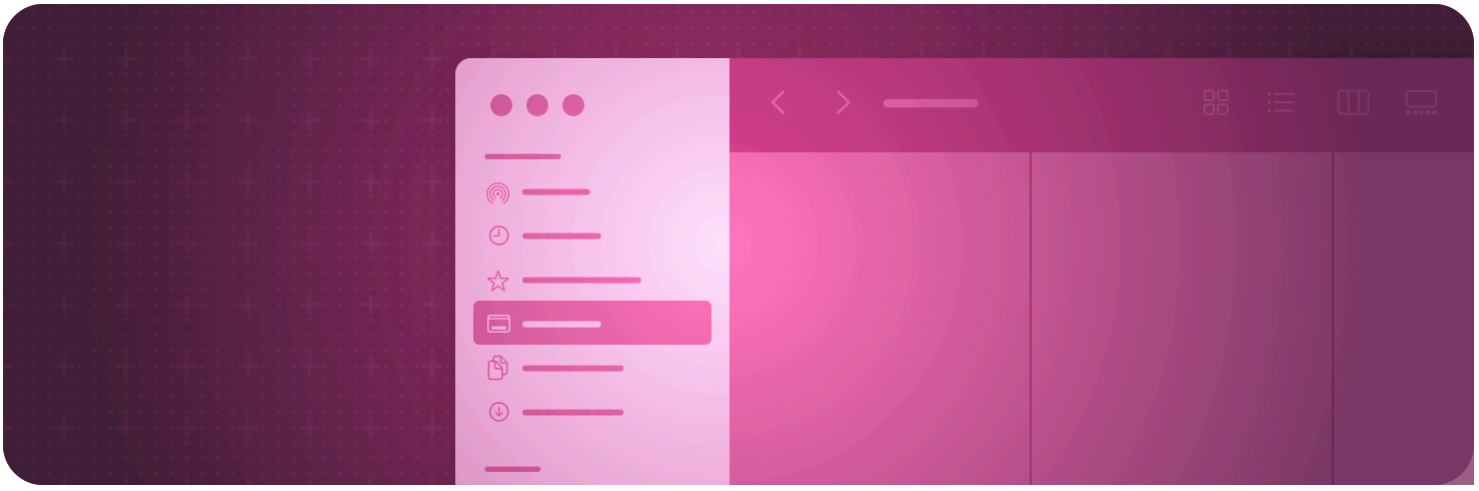
API Collection

Navigation

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

Overview

Use navigation containers to provide structure to your app's user interface, enabling people to easily move among the parts of your app.



For example, people can move forward and backward through a stack of views using a `NavigationStack`, or choose which view to display from a tab bar using a `TabView`.

Configure navigation containers by adding view modifiers like `navigationSplitViewStyle(_:)` to the container. Use other modifiers on the views inside the container to affect the container's behavior when showing that view. For example, you can use `navigationTitle(_:)` on a view to provide a toolbar title to display when showing that view.

Topics

Essentials

Understanding the navigation stack

Learn about the navigation stack, links, and how to manage navigation types in your app's structure.

Presenting views in columns

{} Bringing robust navigation structure to your SwiftUI app

Use navigation links, stacks, destinations, and paths to provide a streamlined experience for all platforms, as well as behaviors such as deep linking and state restoration.

Migrating to new navigation types

Improve navigation behavior in your app by replacing navigation views with navigation stacks and navigation split views.

`struct NavigationSplitView`

A view that presents views in two or three columns, where selections in leading columns control presentations in subsequent columns.

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

Sets the style for navigation split views within this view.

`func navigationSplitViewColumnWidth(CGFloat) -> some View`

Sets a fixed, preferred width for the column containing this view.

`func navigationSplitViewColumnWidth(min: CGFloat?, ideal: CGFloat, max: CGFloat?) -> some View`

Sets a flexible, preferred width for the column containing this view.

`struct NavigationSplitViewVisibility`

The visibility of the leading columns in a navigation split view.

`struct NavigationLink`

A view that controls a navigation presentation.

Stacking views in one column

`struct NavigationStack`

A view that displays a root view and enables you to present additional views over the root view.

`struct NavigationPath`

A type-erased list of data representing the content of a navigation stack.

`func navigationDestination<D, C>(for: D.Type, destination: (D) -> C) -> some View`

Associates a destination view with a presented data type for use within a navigation stack.

`func navigationDestination<V>(isPresented: Binding<Bool>, destination: () -> V) -> some View`

Associates a destination view with a binding that can be used to push the view onto a `NavigationStack`.

`func navigationDestination<D, C>(item: Binding<Optional<D>>, destination: (D) -> C) -> some View`

Associates a destination view with a bound value for use within a navigation stack or navigation split view

Managing column collapse

`struct NavigationSplitViewColumn`

A view that represents a column in a navigation split view.

Setting titles for navigation content

`func navigationTitle(_:)`

Configures the view's title for purposes of navigation, using a string binding.

`func navigationSubtitle(_:)`

Configures the view's subtitle for purposes of navigation.

`func navigationDocument(_:)`

Configures the view's document for purposes of navigation.

`func navigationDocument(_:preview:)`

Configures the view's document for purposes of navigation.

Configuring the navigation bar

`func navigationBarBackButtonHidden(Bool) -> some View`

Hides the navigation bar back button for the view.

`func navigationBarTitleDisplayMode(NavigationBarItem.TitleDisplayMode) -> some View`

Configures the title display mode for this view.

`struct NavigationBarItem`

A configuration for a navigation bar that represents a view at the top of a navigation stack.

Configuring the sidebar

`var sidebarRowSize: SidebarRowSize`

The current size of sidebar rows.

`enum SidebarRowSize`

The standard sizes of sidebar rows.

Presenting views in tabs

{} Enhancing your app's content with tab navigation

Keep your app content front and center while providing quick access to navigation using the tab bar.

`struct TabView`

A view that switches between multiple child views using interactive user interface elements.

`struct Tab`

The content for a tab and the tab's associated tab item in a tab view.

`struct TabRole`

A value that defines the purpose of the tab.

`struct TabSection`

A container that you can use to add hierarchy within a tab view.

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

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

Configuring a tab bar

`func tabViewSidebarHeader<Content>(content: () -> Content) -> some View`

Adds a custom header to the sidebar of a tab view.

`func tabViewSidebarFooter<Content>(content: () -> Content) -> some View`

Adds a custom footer to the sidebar of a tab view.

`func tabViewSidebarBottomBar<Content>(content: () -> Content) -> some View`

Adds a custom bottom bar to the sidebar of a tab view.

`struct AdaptableTabBarPlacement`

A placement for tabs in a tab view using the adaptable sidebar style.

`var tabBarPlacement: TabBarPlacement?`

The current placement of the tab bar.

`struct TabBarPlacement`

A placement for tabs in a tab view.

`var isTabBarShowingSections: Bool`

A Boolean value that determines whether a tab view shows the expanded contents of a tab section.

`struct TabBarMinimizeBehavior`

`enum TabViewBottomAccessoryPlacement`

A placement of the bottom accessory in a tab view. You can use this to adjust the content of the accessory view based on the placement.

Configuring a tab

`func sectionActions<Content>(content: () -> Content) -> some View`

Adds custom actions to a section.

`struct TabPlacement`

A place that a tab can appear.

`struct TabContentBuilder`

A result builder that constructs tabs for a tab view that supports programmatic selection. This builder requires that all tabs in the tab view have the same selection type.

`protocol TabContent`

A type that provides content for programmatically selectable tabs in a tab view.

`struct AnyTabContent`

Type erased tab content.

Enabling tab customization

`func tabViewCustomization(Binding<TabViewCustomization>?) -> some View`

Specifies the customizations to apply to the sidebar representation of the tab view.

`struct TabViewCustomization`

The customizations a person makes to an adaptable sidebar tab view.

`struct TabCustomizationBehavior`

The customization behavior of customizable tab view content.

Displaying views in multiple panes

`struct HSplitView`

A layout container that arranges its children in a horizontal line and allows the user to resize them using dividers placed between them.

`struct VSplitView`

A layout container that arranges its children in a vertical line and allows the user to resize them using dividers placed between them.

Deprecated Types

~~`struct NavigationView`~~

A view for presenting a stack of views that represents a visible path in a navigation hierarchy.

Deprecated

~~`func tabItem<V>(() -> V) -> some View`~~

Sets the tab bar item associated with this view.

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.

- 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.

- 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.