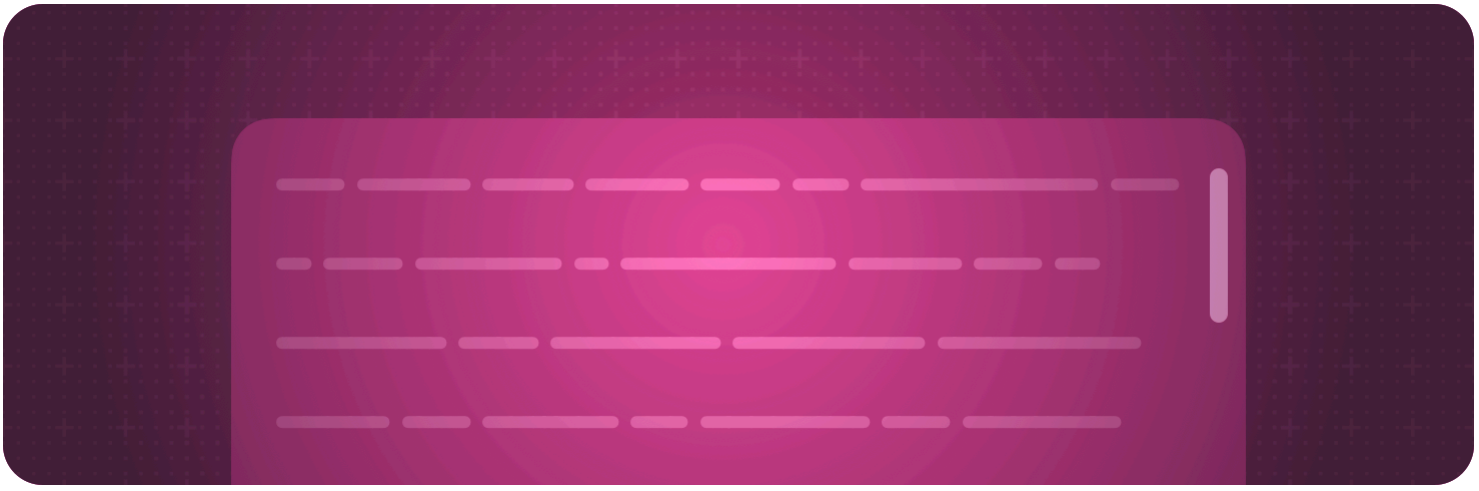
API Collection

Scroll views

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

Overview

When the content of a view doesn't fit in the display, you can wrap the view in a `ScrollView` to enable people to scroll on one or more axes. Configure the scroll view using view modifiers. For example, you can set the visibility of the scroll indicators or the availability of scrolling in a given dimension.



You can put any view type in a scroll view, but you most often use a scroll view for a layout container with too many elements to fit in the display. For some container views that you put in a scroll view, like lazy stacks, the container doesn't load views until they are visible or almost visible. For others, like regular stacks and grids, the container loads the content all at once, regardless of the state of scrolling.

Lists and Tables implicitly include a scroll view, so you don't need to add scrolling to those container types. However, you can configure their implicit scroll views with the same view modifiers that apply to explicit scroll views.

For design guidance, see Scroll views in the Human Interface Guidelines.

Topics

Creating a scroll view

`struct ScrollView`

A scrollable view.

`struct ScrollViewReader`

A view that provides programmatic scrolling, by working with a proxy to scroll to known child views.

`struct ScrollViewProxy`

A proxy value that supports programmatic scrolling of the scrollable views within a view hierarchy.

Managing scroll position

`func scrollPosition(Binding<ScrollPosition>, anchor: UnitPoint?) -> some View`

Associates a binding to a scroll position with a scroll view within this view.

`func scrollPosition(id: Binding<(some Hashable)?>, anchor: UnitPoint?) -> some View`

Associates a binding to be updated when a scroll view within this view scrolls.

`func defaultScrollAnchor(UnitPoint?) -> some View`

Associates an anchor to control which part of the scroll view's content should be rendered by default.

`func defaultScrollAnchor(UnitPoint?, for: ScrollAnchorRole) -> some View`

Associates an anchor to control the position of a scroll view in a particular circumstance.

`struct ScrollAnchorRole`

A type defining the role of a scroll anchor.

`struct ScrollPosition`

A type that defines the semantic position of where a scroll view is scrolled within its content.

Defining scroll targets

`func scrollTargetBehavior(some ScrollTargetBehavior) -> some View`

Sets the scroll behavior of views scrollable in the provided axes.

`func scrollTargetLayout(isEnabled: Bool) -> some View`

Configures the outermost layout as a scroll target layout.

`struct ScrollTarget`

A type defining the target in which a scroll view should try and scroll to.

`protocol ScrollTargetBehavior`

A type that defines the scroll behavior of a scrollable view.

`struct ScrollTargetBehaviorContext`

The context in which a scroll target behavior updates its scroll target.

`struct PagingScrollTargetBehavior`

The scroll behavior that aligns scroll targets to container-based geometry.

`struct ViewAlignedScrollTargetBehavior`

The scroll behavior that aligns scroll targets to view-based geometry.

`struct AnyScrollTargetBehavior`

A type-erased scroll target behavior.

`struct ScrollTargetBehaviorProperties`

Properties influencing the scroll view a scroll target behavior applies to.

`struct ScrollTargetBehaviorPropertiesContext`

The context in which a scroll target behavior can decide its properties.

Animating scroll transitions

`func scrollTransition(ScrollTransitionConfiguration, axis: Axis?, transition: (EmptyVisualEffect, ScrollTransitionPhase) -> some VisualEffect) -> some View`

Applies the given transition, animating between the phases of the transition as this view appears and disappears within the visible region of the containing scroll view.

`func scrollTransition(topLeading: ScrollTransitionConfiguration, bottomTrailing: ScrollTransitionConfiguration, axis: Axis?, transition: (EmptyVisualEffect, ScrollTransitionPhase) -> some VisualEffect) -> some View`

Applies the given transition, animating between the phases of the transition as this view appears and disappears within the visible region of the containing scroll view.

`enum ScrollTransitionPhase`

The phases that a view transitions between when it scrolls among other views.

`struct ScrollTransitionConfiguration`

The configuration of a scroll transition that controls how a transition is applied as a view is scrolled through the visible region of a containing scroll view or other container.

Responding to scroll view changes

`func onScrollGeometryChange<T>(for: T.Type, of: (ScrollGeometry) -> T, action: (T, T) -> Void) -> some View`

Adds an action to be performed when a value, created from a scroll geometry, changes.

`func onScrollTargetVisibilityChange<ID>(idType: ID.Type, threshold: Double, ([ID]) -> Void) -> some View`

Adds an action to be called with information about what views would be considered visible.

`func onScrollVisibilityChange(threshold: Double, (Bool) -> Void) -> some View`

Adds an action to be called when the view crosses the threshold to be considered on/off screen.

`func onScrollPhaseChange(_:)`

Adds an action to perform when the scroll phase of the first scroll view in the hierarchy changes.

`struct ScrollGeometry`

A type that defines the geometry of a scroll view.

`enum ScrollPhase`

A type that describes the state of a scroll gesture of a scrollable view like a scroll view.

`struct ScrollPhaseChangeContext`

A type that provides you with more content when the phase of a scroll view changes.

Showing scroll indicators

`func scrollIndicatorsFlash(onAppear: Bool) -> some View`

Flashes the scroll indicators of a scrollable view when it appears.

`func scrollIndicatorsFlash(trigger: some Equatable) -> some View`

Flashes the scroll indicators of scrollable views when a value changes.

`func scrollIndicators(ScrollIndicatorVisibility, axes: Axis.Set) -> some View`

Sets the visibility of scroll indicators within this view.

`var horizontalScrollIndicatorVisibility: ScrollIndicatorVisibility`

The visibility to apply to scroll indicators of any horizontally scrollable content.

`var verticalScrollIndicatorVisibility: ScrollIndicatorVisibility`

The visiblity to apply to scroll indicators of any vertically scrollable content.

`struct ScrollIndicatorVisibility`

The visibility of scroll indicators of a UI element.

Managing content visibility

`func scrollContentBackground(Visibility) -> some View`

Specifies the visibility of the background for scrollable views within this view.

`func scrollClipDisabled(Bool) -> some View`

Sets whether a scroll view clips its content to its bounds.

`struct ScrollContentOffsetAdjustmentBehavior`

A type that defines the different kinds of content offset adjusting behaviors a scroll view can have.

Disabling scrolling

`func scrollDisabled(Bool) -> some View`

Disables or enables scrolling in scrollable views.

`var isScrollEnabled: Bool`

A Boolean value that indicates whether any scroll views associated with this environment allow scrolling to occur.

Configuring scroll bounce behavior

```
func scrollBounceBehavior(ScrollBounceBehavior, axes: Axis.Set) -> some View
```

Configures the bounce behavior of scrollable views along the specified axis.

```
var horizontalScrollBounceBehavior: ScrollBounceBehavior
```

The scroll bounce mode for the horizontal axis of scrollable views.

```
var verticalScrollBounceBehavior: ScrollBounceBehavior
```

The scroll bounce mode for the vertical axis of scrollable views.

```
struct ScrollBounceBehavior
```

The ways that a scrollable view can bounce when it reaches the end of its content.

Configuring scroll edge effects

```
func scrollEdgeEffectStyle(ScrollEdgeEffectStyle?, for: Edge.Set) -> some View
```

Configures the scroll edge effect style for scroll views within this hierarchy.

```
struct ScrollEdgeEffectStyle
```

A structure that defines the style of pocket a scroll view will have.

```
func safeAreaBar(edge:alignment:spacing:content:)
```

Shows the specified content as a custom bar beside the modified view.

Interacting with a software keyboard

```
func scrollDismissesKeyboard(ScrollDismissesKeyboardMode) -> some View
```

Configures the behavior in which scrollable content interacts with the software keyboard.

```
var scrollDismissesKeyboardMode: ScrollDismissesKeyboardMode
```

The way that scrollable content interacts with the software keyboard.

```
struct ScrollDismissesKeyboardMode
```

The ways that scrollable content can interact with the software keyboard.

Managing scrolling for different inputs

`func scrollInputBehavior(ScrollInputBehavior, for: ScrollInputKind) -> some View`

Enables or disables scrolling in scrollable views when using particular inputs.

`struct ScrollInputKind`

Inputs used to scroll views.

`struct ScrollInputBehavior`

A type that defines whether input should scroll a view.

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.

- Custom layout

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

- 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.